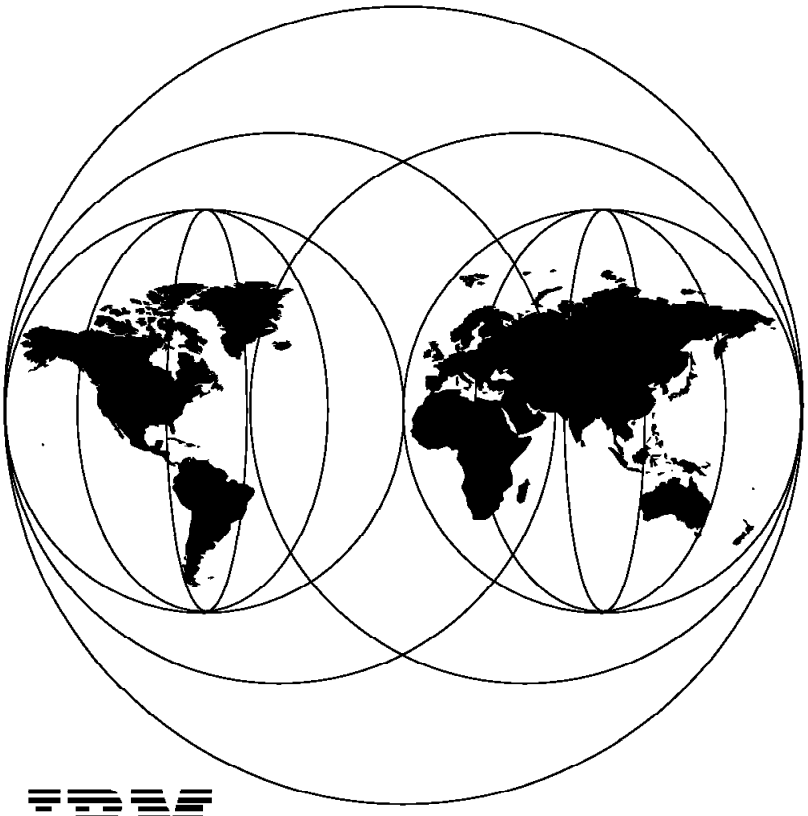
International Technical Support Organization

SG24-4170-01

Multiprotocol Transport Networking (MPTN) Architecture Tutorial and Product Implementations

November 1995



IBM

International Technical Support Organization
Raleigh Center

International Technical Support Organization

SG24-4170-01

Multiprotocol Transport Networking (MPTN) Architecture Tutorial and Product Implementations

November 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

Second Edition (November 1995)

This edition applies to V4R3 of VTAM, Program Number 5695-117 for use with MVS/ESA and Communications Manager/2 V1.1, Program Number 5871-Z72 for use with OS/2 V2.0.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. HZ8, Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Abstract

This publication is intended to help customers, IBM networking systems specialists and representatives working in a mixed-protocol networking environment to gain an understanding of the Multiprotocol Transport Networking (MPTN) architecture developed by IBM. This publication is structured to provide readers with easy access to an overview of the architecture as well as more detailed in-depth technical information. The AnyNet product family with its first deliverables, is also discussed in the context of functionallity provided and conformance to the MPTN architecture.

The general reader need only have a basic understanding of the networking principles and architectures, in particular, SNA APPN and TCP/IP, although a more detailed knowledge is presumed for the more comprehensive chapters.

(139 pages)

Contents

Figures

Tables

Special Notices

This publication provides an overview and tutorial of the Multiprotocol Transport Networking (MPTN) architecture developed by IBM as well as information on the AnyNet product family. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM VTAM V4R3 for MVS/ESA and AnyNet/MVS or AnyNet/2. See the PUBLICATIONS SECTION of the IBM Programming Announcement for those products listed above, for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
AnyNet	APPN
AS/400	CICS
DATABASE 2	DB2
Distributed Relational Database Architecture	DRDA
IBM	IMS/ESA
MVS/ESA	NetView
OS/2	OS/400
VTAM	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Windows is a trademark of Microsoft Corporation.

IPX is a trademark of Novell, Inc.

AppleTalk	Apple Corporation
DEC, DECnet	Digital Equipment Corporation
Novell, IPX	Novell, Inc.
OSF	Open Software Foundation, Inc.
Sun, Network File System, NFS	Sun Microsystems, Inc.
Microsoft, Windows	Microsoft Corporation
X-Windows	Massachusetts Institute of Technology
X Window System	Massachusetts Institute of Technology

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to provide an introductory overview as well as a tutorial-style description of the multiprotocol transport networking (MPTN) architecture developed by IBM. This document is structured so that the first two chapters will provide the reader with a general understanding of the issues involved in interconnecting mixed-protocol networks. MPTN is positioned with respect to other current internetworking solutions and a description is provided of the significant features and benefits of the MPTN approach in various networking scenarios.

Following these two introductory chapters, the reader is taken through the details of the major components which make up an MPTN network, that is, the access node, gateway node and address mapper. The level of detail provided is sufficient for the reader to grasp the issues involved in providing such a generalized mixed-protocol networking solution together with an understanding of how these components interoperate to provide an overall system.

Finally, this document includes a description of IBM's current AnyNet product offerings which implement the MPTN architecture. It discusses the functions provided by these products and looks at their implementation particularly in relation to the MPTN architecture.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, "Introduction"

  This chapter provides an introduction into the concept of multiprotocol networking and the solutions that exist today. MPTN is contrasted with these as an alternative solution and its particular advantages are explained as a software solution in existing network nodes rather than a hardware solution using additional network nodes, is provided here. We also look at MPTN in terms of the IBM Networking Blueprint.

- Chapter 2, "Architecture Overview"

  This chapter provides an overall picture of the MPTN architecture and of its major components. We look at addressing and the problems associated with different address formats in a mixed-protocol networking environment and also at the mechanisms involved in establishing MPTN connections.

- Chapter 3, "MPTN Transport Services"

  This chapter describes the transport-layer protocol boundary (TLPB) as an implementation of the CTS described in the Networking Blueprint. Here we look at different connection types and the transport characteristic requirements of varying protocols. We discuss what functions MPTN must address in order to satisfy these varied requirements for the generalized case. We also look at the semantics involved at the TLPB and how these need relate to specific transport characteristics.

- Chapter 4, "MPTN Access Node"

  This chapter describes the MPTN access node and its structure. We discuss what constitutes a user of the TLPB and how various transport provider

protocols interface with the TLPB. We look at the various components of the TLPB and how protocol compensation is achieved between non-matching transport users and transport providers. We look at connection establishment and how data is routed through the access node for both the native and nonnative cases.

- Chapter 5, “MPTN Gateway Node”

  This chapter describes the MPTN gateway node and its structure. We look at the various mixed network configurations that MPTN must address and discuss the approach inherent within MPTN to accommodate these internetworking requirements. We look at the protocols between MPTN end nodes and gateway nodes as well as the protocols between gateway nodes. We discuss the function of the MPTN gateway manager and how routing is achieved, resources located, paths selected, and connections established.

- Chapter 6, “MPTN Address Mapping Services”

  This chapter describes the functions of address mapping within an MPTN environment and an address mapping services model is described. We discuss the architecture of the MPTN address mapper function, how addresses are registered, partners located and connections established. We discuss the particular requirements of address management and look at three approaches to providing address mapping services for nonnative transport users and providers.

- Chapter 7, “MPTN Product Implementations”

  This chapter describes the current MPTN implementations, namely the current members of the AnyNet product family. In this chapter we discuss implementations of the MPTN Access Node architecture on the following platforms: VTAM, OS/2, OS/400, AIX/6000 and Windows, and the MPTN Gateway Node architecture on the following platforms: VTAM and OS/2. We discuss the structure of the products, how network calls are mapped (naming and addressing) and how calls are set up.

- Chapter 8, “ MPTN Network Management”

  This chapter briefly describes the implications for network management that MPTN places upon users and network solutions providers. We discuss the approach IBM has adopted in general as well as for the AnyNet network cases, including management of the MPTN gateway. Finally, we discuss the future directions for mixed-protocol network management.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Systems Network Architecture Technical Overview*, GC30-3073
- *VTAM Multiprotocol Transport Feature: Sockets over SNA User's Guide*, SC31-6487
- *VTAM Multiprotocol Transport Feature: APPC over TCP/IP User's Guide*, SC31-6488
- *Networking Blueprint: Executive Overview*, GC31-7057
- *Multiprotocol Transport Network (MPTN) Architecture: Technical Overview*, GC31-7073
- *Multiprotocol Transport Network (MPTN) Architecture: Formats*, GC31-7074(This publication is available in CD-ROM only on SK2T-2620 and SK2T-8515.)

International Technical Support Organization Publications

- *Local Area Network Concepts and Products*, GG24-3178
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *APPC and CPI-C Product Implementations*, GG24-3520
- *VTAM V3R4.2 AnyNet/MVS Implementation*, GG24-4066

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser to the following URL:

`http://www.redbooks.ibm.com/redbooks`

IBM employees may access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks Home Page at the following URL:

`http://w3.itsc.pok.ibm.com/redbooks/redbooks.html`

Send comments via E-mail to: `redbooks@vnet.ibm.com`

Acknowledgements

The author of the second edition of this document is:

Chris Mason IBM Belgium

The advisor for this project was:

Peter Lenhard International Technical Support Organization, Raleigh Center

The authors of the first edition of this document are:

Andreas Hirschbold IBM Germany

Peter Ieraci IBM Australia

The advisor for the first edition of this document was:

Peter Lenhard International Technical Support Organization, Raleigh Center

This publication is the result of a residency conducted at the International Technical Support Organization, Raleigh Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

John Fetvedt IBM Research Triangle Park

Paul Golick IBM Research Triangle Park

Matthew Hess IBM Research Triangle Park

Sandy Hobgood IBM Research Triangle Park

Leah Ketring IBM Research Triangle Park

Alisa Morse IBM Research Triangle Park

Kimberly Patraw IBM Research Triangle Park

Diane Pozefsky IBM Research Triangle Park

Kathleen Riordan IBM Research Triangle Park

Steve Smith IBM Research Triangle Park

Karen Tracey IBM Research Triangle Park

Roger Turner IBM Research Triangle Park

Walter Wheeler IBM Research Triangle Park

Chapter 1. Introduction

Multiprotocol transport networking (MPTN), developed by IBM, is an open architecture for:

- Mixed-protocol networking: running applications associated with one network protocol over different network protocols
- Network concatenation: connecting matching applications across networks of different protocols through gateways

Throughout this document we will be discussing MPTN functions and operations from an architectural viewpoint. We will explore the structure and features of such a networking architecture together with some practical operational scenarios. We will explain the necessary functions required of the access node, address mapping node and gateway node and how these work together to interconnect mixed-protocol networks. We will show where MPTN splits the communications protocol stack so that application programs can operate over transport protocols for which they were not originally intended to operate.

The reader is cautioned to bear in mind that, in general, we are not discussing specific product implementations (apart from Chapter 7, “MPTN Product Implementations” on page 91) but are looking at MPTN concepts and principles of operation.

This introductory chapter gives a brief overview of the requirements present in today’s growing networks which require a solution for mixed-protocol networking. We will look at the potential benefits of a software based solution together with reasons for such an approach. We will also give a general overview of current multiprotocol solutions, relating them to the MPTN architecture, as well as some network examples and solutions.

1.1 Requirements for Mixed-Protocol Networking

With the growth of networking in general and local area networks in particular, many large customer networks have become confederations of individual networks running different networking protocols. This heterogeneity has arisen for a number of reasons. Some of these include:

- Shift of customer interest away from selecting a particular networking architecture in favor of finding solutions to business problems, regardless of specific network protocol required
- Inter-enterprise information exchange requirement, for example, direct manufacturing, order placement, or billing systems
- Interconnection of decentralized departmental systems which often use different networking protocols
- Company mergers requiring interconnection

It is not uncommon today to see customer configurations with dozens of networks running as many as four or five different network protocols, such as SNA, TCP/IP, NetBIOS, IPX, DEC, or AppleTalk. The problem will become worse as networks continue to expand and companies increasingly wish to communicate externally.

As a result of these conditions we are seeing a need to provide a generalized solution to the problem of interconnecting networks running different protocols. By doing so, we will satisfy the customer′s need to identify the best application solution irrespective of network transport considerations, as mixed-protocol networking will allow an application associated with one protocol to run without change over a different networking protocol.

1.2 Benefits

Applications that operate over one network typically cannot interoperate with applications on other networks. Today, the customer circumvents this problem by writing application gateways using remote APIs or encapsulation (see 1.6, “Positioning MPTN to Other Multiprotocol Solutions” on page 5) for those programs for which interoperability is important. The main drawback with this approach is that each gateway handles only one specific application, requiring n*(n-1)/2 gateways to provide full application connectivity, where n represents the number of different applications. Let′s see what this formula really means: If the application runs in 3 network environments, 3 gateways will be required. Not too bad so far! If the application runs in 4 network environments, 6 gateways will be required. Not looking so good! If the application runs in 5 network environments, 10 gateways will be required. This is clearly becoming too much! This can obviously be very expensive and hence become economically unjustifiable.

However, if the APIs used by the application programs are made independent of the underlying network protocol, then communication across connected networks would be much simpler, hence more cost effective, and would solve the problem for most customers.

We can summarize the benefits that a cost-effective mixed-protocol networking solution can provide to customers as follows:

- Ability to add a new application program based solely on how well it meets business needs without requiring the network administrator to install a network running a specific protocol, including additional software in existing nodes and, perhaps, additional nodes for routing purposes. Off-the-shelf application programs can now be used in new environments for which they were not designed.
- Access to application support services that are not available over currently installed networking protocols. Examples include DRDA (Distributed Relational Database Architecture) over TCP, FTAM over SNA, CPI-C over TCP/IP, and OSF DCE over SNA.
- Ability to change from a network of one protocol to that of another while maintaining the investment in existing application programs.
- Ability to simplify networks by reducing the number of network protocols being run without needing to rewrite the large number of existing installed applications.
- Ability to exchange information with other organizations (vendors, suppliers, other departments, subsidiaries, etc.) without installing additional networking protocols.
- Ability to develop new application programs on an application programming interface (API) based solely on the function of that API, not being limited to an API that accompanies the installed network.

- Ability to develop applications for many different network protocols without duplicate development expenses. As described above, this requires an application interface to networks providing the services required by the application which are truly independent of the network protocol.

1.3 Why a Transport Solution?

The problem is as follows: “There is a need to provide a means whereby applications will have the ability to communicate with other applications, typically written to the same API, over a network protocol different from that over which the API for such applications was designed to work.”

The proposed solution is: “Arrange that there is a change of protocol over the transport layer of the underlying network protocol.”

The question arises: “Why is this transport-based solution the best one available?”

The reasons are the following:

- The interface between the transport layer and the session layer is the natural boundary between services oriented to application functions and services oriented to network functions. Everything above this interface is a partner-to-partner protocol which assumes, for connection-oriented communication, reliable delivery. Everything below this interface is concerned with passing data through the network.
- Some existing network protocols, such as TCP/IP and NetBIOS, have a natural boundary at the transport layer, in the form of APIs, making this a logical place to select as the interface between applications and networks.
- There are many similarities in the transport functions provided by different protocol stacks, making it practical both to define general services and to specify the mappings between two protocols. *This is the highest point in the protocol stack where this is true.*

  The commonality between protocols can be exploited to standardize a small number of adjustments to the functions supported to provide the functions required. These “adjustments” are formally called *compensations* in MPTN. Thus the compensations provide a toolkit from which a particular protocol combination can be built. Most of these compensations are very simple and involve adding, at most, a label and a length to the data provided by the transport user before sending it over the transport provider.
- By making changes in the logic supporting the communications protocol within an operating system platform, it is possible to adapt to a different network protocol while leaving application programs unchanged. Clearly this is much more economical than changing the applications themselves since there are many more applications than operating system platforms.

1.4 MPTN and X/Open

X/Open has also identified the need for mixed-protocol networking and has proposed a general solution. X/Open has already defined an interface to networks, the X/Open Transport Interface (XTI), and shown how this interface maps to four different network types (OSI transport, TCP/UDP transport, SNA and NetBIOS). However, XTI is limited. True transport independence with XTI can

only be achieved today if the application respects the semantics of the API offered by XTI and does not make any assumptions about the transport it uses. This is explained in Appendix C of the XTI specification. In other words, XTI on its own cannot guarantee true transport independence unless the application is restricted to the least common denominator subset that is supported by all protocols.

This, then, is the difference between the approach taken by XTI and MPTN to mixed-protocol networking. X/Open defined a new interface (XTI) for which there were no preexisting applications, whereas MPTN allows existing applications to function without change.

X/Open has accepted and published the IBM contribution of MPTN for adoption as a public standard for mixed-protocol internetworking.

1.5 Relationship to the Networking Blueprint

Support for multiple protocols can be achieved in a number of ways and the Networking Blueprint illustrated in Figure 1 on page 5 provides the context for discussing them.

The common transport semantics (CTS) layer in the Networking Blueprint provides support for multiple protocols at the transport layer. CTS is derived from the semantics of all of the major protocols' transport stacks. If needed functions are missing from any of the transport providers, CTS itself provides those functions.

CTS function can be achieved in different ways depending upon the situation:

1. CTS encompasses the normal case where the protocols of the application match those of the transport network. This is shown by the Same Protocol path in Figure 1 on page 5.
2. CTS function can be delivered by existing industry standards, such as RFC 1006 for OSI over TCP/IP or RFCs 1001 and 1002 for NetBIOS over TCP/IP. This is shown by the Existing Standards path in Figure 1 on page 5.
3. The formats and protocols of MPTN deliver CTS function in situations where the native protocol of the application does not match the transport network. Compensation for missing functions is provided in those situations. For example, MPTN defines how SNA can be the transport network for sockets applications. This is shown by the MPTN path in Figure 1 on page 5.

CTS solutions allow customers to run a wider variety of applications without having to install additional transport services and the network services on which they depend.

Support for multiple protocols can also be achieved outside of CTS, such as through subnetwork technologies like local area networks and frame-relay or multiprotocol routers like the IBM 6611 Network Processor. These solutions play an important complementary role to the CTS solutions.

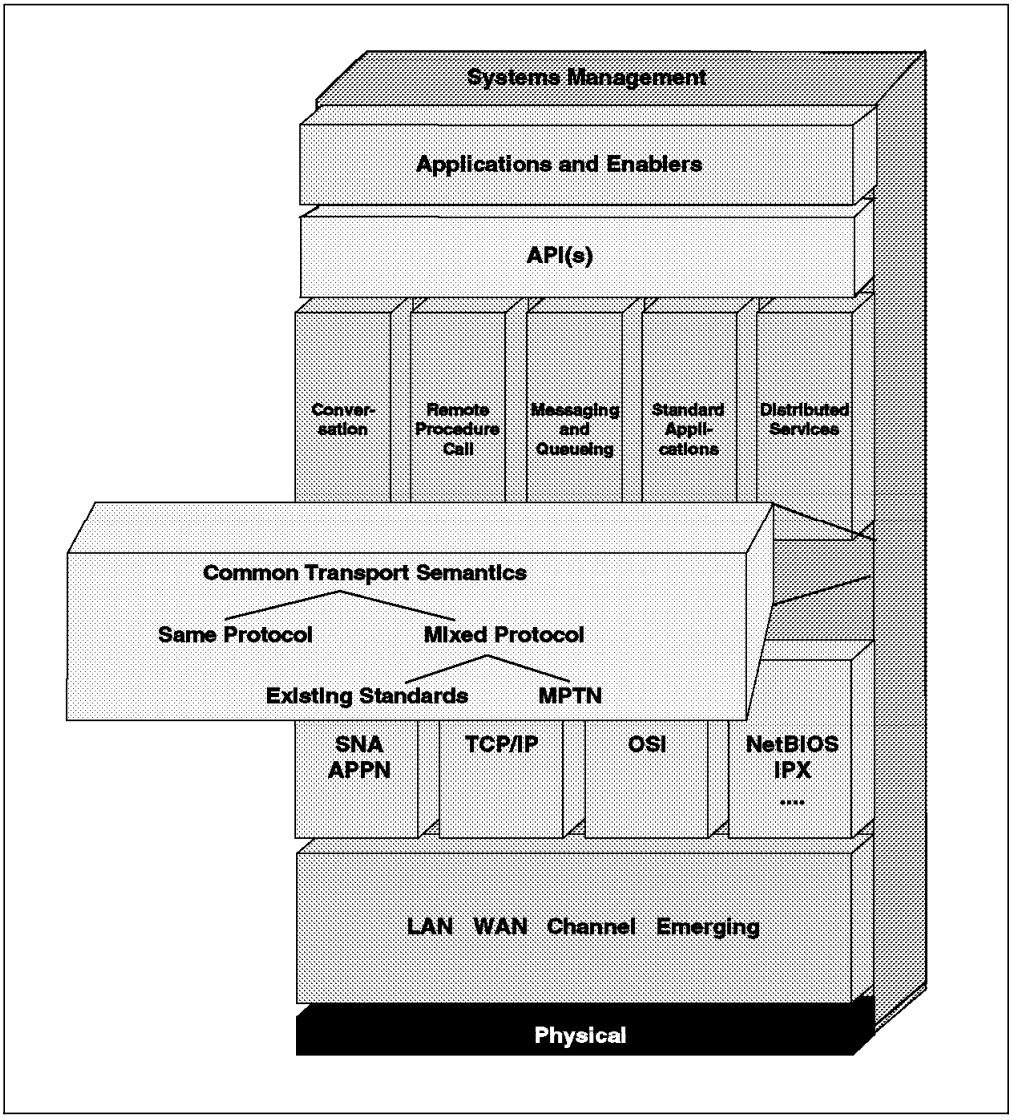


Figure 1. Relationship of the MPTN Architecture to the Networking Blueprint CTS Layer

1.6 Positioning MPTN to Other Multiprotocol Solutions

In this section we will look at how MPTN implements a solution to multiprotocol networking by first considering the various methods used to date. We will briefly describe some of these (illustrated in Figure 2 on page 6) and then compare them to MPTN.

- **Application gateways** connect two applications cooperating in performing the same function (for example, mail exchange) but running over different network protocols (for example, TCP/IP and OSI). These two applications are dependent on the underlying network protocol and, consequently, the application gateway solution is dependent on the underlying network protocols. Moreover it is specific for a pair of applications. Where the two applications run over different network protocols, as in the popular example given, electronic mail, this is the only solution possible.
- **Middleware** solutions often include the support of new application program interface (API) that is intended to shield the programmers from the

complexities of communications. This makes it easier to support connectivity over a wide choice of transport protocols. There are a multitude of considerations to be taken into account in determining the best middleware type of approach. The only factor being pointed out here is that, by relying on the use of a specific (new) API, such a solution does nothing to enable existing applications to operate over a range of transport protocols.

- **XTI (X/Open Transport Interface)** is a special middleware solution developed by X/Open. It is an API for accessing OSI, TCP/IP, SNA, and NetBIOS transports. XTI, like the mixed-protocol RFCs (1001 and 1002 for NetBIOS over TCP/IP and 1006 for OSI over TCP/IP), is an example of limited, but useful, efforts to allow for mixed-protocol network access.

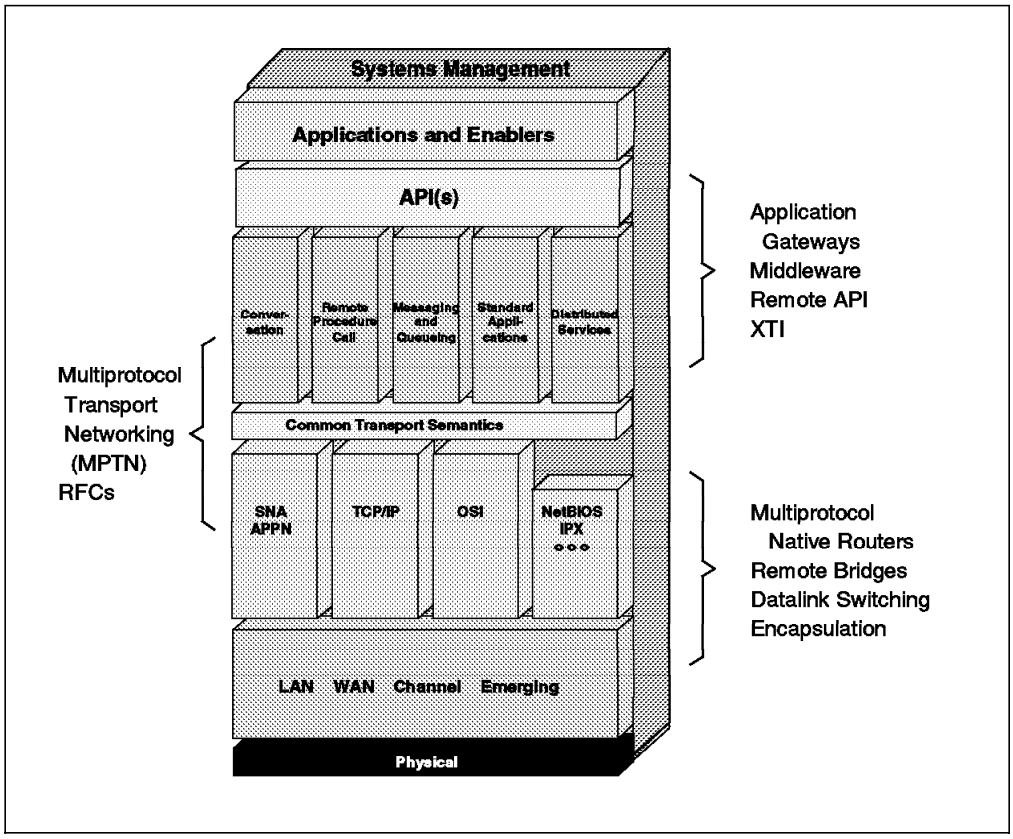


Figure 2. Multiprotocol Techniques - Relationship to Networking Blueprint

- **Remote API** solutions can be developed to support existing APIs and, thus, applications. However, they require dedicated support for each emulated API. Since there may be many APIs, some limitations are usually introduced. They depend on two nodes, the client node where the remote API runs and server node which simulates the presence of the application running actually on the remote client node. This dependency on two nodes means that the data path between two clients actually passes through a third node, the server node.
- **Multiprotocol routers** operate in the *transport* portion of the Networking Blueprint (that is, at ISO layer 3). Routers are a very popular approach to multiprotocol networking because they are meant to adapt to whatever protocols exist in the networks being interconnected. The difficulty associated with this type of solution is that routers do not reduce the number of protocols in the overall network. Indeed multiprotocol routers may add yet another network protocol, typically IP, to the existing network protocols.

Even though they can enable consolidation of physical resources in the backbone, typically wide area, network, the exterior networks (and their workstations or end nodes) continue to require full support for all protocols that their applications require.

- **Remote Bridges** are used to extend LANs via bridges, by using bridges connected via non-LAN facilities (leased-lines, switched-lines, X.25, etc.).

  The essential feature of this technique is that the frames are taken from the LAN at the MAC layer (sub-layer of ISO layer 2). These are then sent as data across the wide area network where they are reconstituted as MAC frames on the target LAN.

  The aspect of remote bridging of significance here is the method used in the wide area network for connecting the exterior LANs.

  The typical method used is that of *encapsulation*, whereby a transport connection is provided across an intermediate transport network. In this case, there is a transport session between the remote bridges which multiplexes all of the traffic between the two LANs.

  An example of this is to use frame relay as the intermediate transport network, which functions as a subnetwork, but operates over a wide area network. It routes the LAN packets as *pure* layer 2 packets, thus avoiding the overhead of processing layers 3 and 4.

- **Encapsulation** encompasses a whole range of possibilities, as it can be used to describe many aspects and/or methods of mixed-protocol networking. In the current context, it is limited to those techniques used for *tunneling* through one (transport) network. From the viewpoint of the *supported* network, the *supporting* transport network is being used as a *link*. From the viewpoint of the *supporting* network, the *supported* network*'*s *link* is just another *user session* between end points within the network.

  Encapsulation involves taking the (usually link-level) packets of the *supported* network and sending them as data on a session through the *supporting* network. They are de-encapsulated at the other session end-point and reconstituted as link-level packets. The emulated link can be used to connect end nodes across a single transport network, as in a *single network configuration*.

  When encapsulation (or de-encapsulation) is performed in a single gateway linking two different transport networks (a single gateway configuration), it is required that, in place of a complementary gateway, the end node performs the complementary (de-)encapsulation.

  In a double gateway configuration, the end nodes are unaffected. They communicate over native transport protocols to the gateways, where the encapsulation and link emulation is done across, typically, a wide area network.

  The distinguishing characteristics of this approach is that both of the protocol stacks are present in the systems where the encapsulation and de-encapsulation are done. This may be at the end nodes and/or at the gateway(s), depending on the particular configuration.

  Typically, encapsulation techniques incorporate some methods to filtering traffic, to eliminate dependency on timing or to eliminate excessive broadcasts.

  Another aspect of this approach is that it inherently allows for the multiplexing of many user applications over the emulated link.

- **Datalink Switching** is a special case of encapsulation. It has been taken to Internet as a standard for encapsulating SNA and NetBIOS over TCP/IP. It involves sending link-level packets (usually as data over a transport session) through a backbone network. It goes beyond encapsulation in that some of the packets are intercepted and used to emulate specific aspects of the transport protocol. Specifically, this is used to filter out some LAN broadcast messages to reduce traffic on the backbone network. Since these packets are being suppressed, it is necessary to compensate for the function lost, to reproduce their effect for the endpoint(s). One of the more important functions of these broadcasts is to locate session partners. The datalink switching component thus must capture and maintain data about the requests, and the location of partners, so session connections can be emulated across the backbone network.

  Another important aspect of this technique is that the data link control (DLC) of the exterior network(s) is *terminated*. That is, the gateway handles any timing-dependent traffic between itself and the end station (for example, keep-alive messages or polling). This aspect of datalink switching results from the difficulties of bridging LAN protocols across (usually slower) WANs.

  Note that MPTN also terminates the DLC and provides connections across the backbone network. The major difference is that MPTN is *architected* to provide this in a standard manner for multiple protocols, whereas datalink switching implementations are protocol-specific.

- The essence of the **multiprotocol transport networking (MPTN)** technique is that it allows an application (or the upper layers supporting the application) to communicate over a transport network for which it was not designed. This is done in a manner that is *totally transparent to the application*, by provision of complete compensations for missing functions in each transport protocol. Provision of this capability allows nonnative applications to communicate end-to-end.

- Running **parallel networks** is the only solution when a network has implemented none of the above techniques.

1.6.1 MPTN Internetworking

Traditional internets require that every attached end node must implement the prescribed internetworking protocol to gain connectivity beyond their local subnets. Such internets are called single-protocol transport networks (SPTNs) in MPTN terminology, and it is obvious that SPTNs using different protocols cannot interoperate. This is illustrated in Figure 3 on page 9.

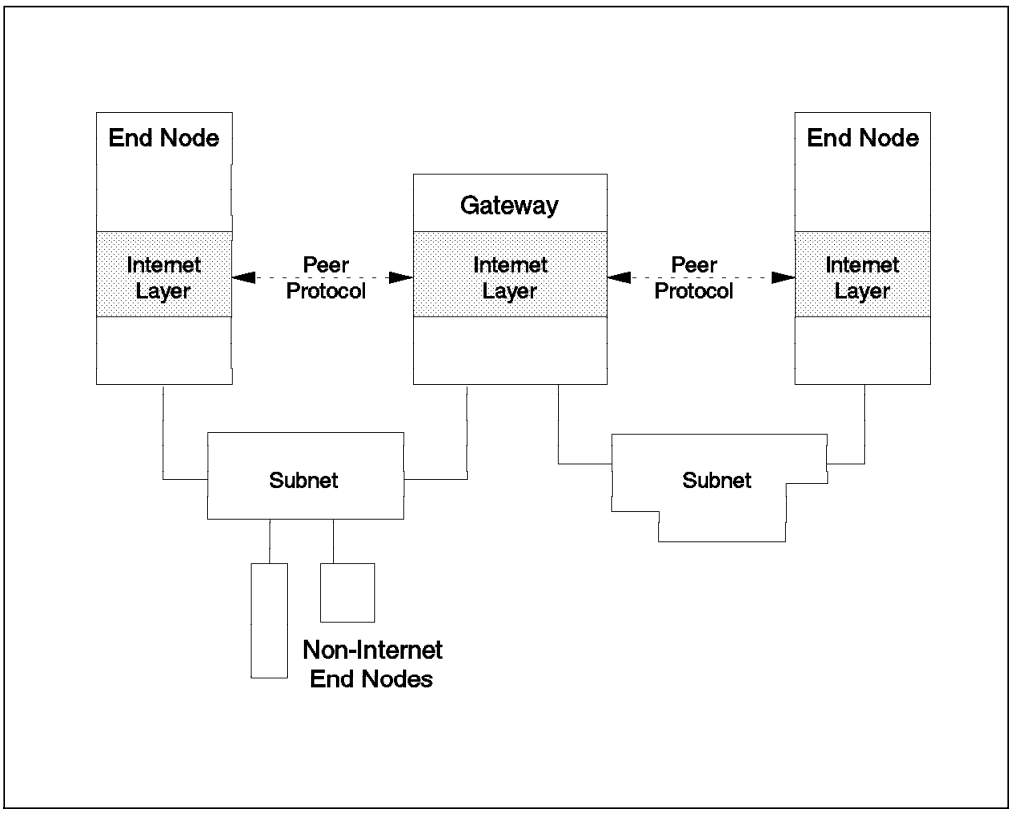


Figure 3. SPTN Structure. Communication between end nodes without the prescribed internet protocol is confined to their respective local subnet.

As illustrated in Figure 4, there are two sets of relationships that are important to the MPTN architecture: peer relationships, and relationships between functions that use the transport network and functions that provide transport services.

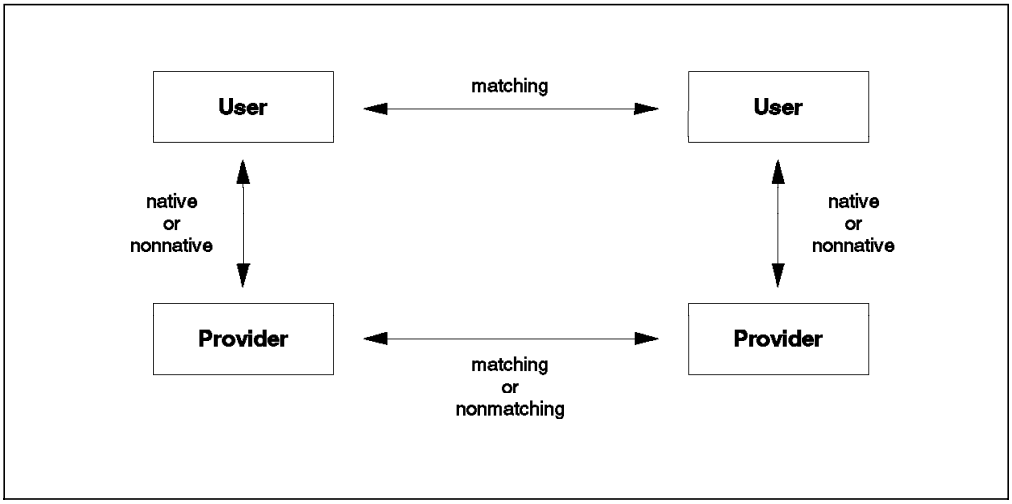


Figure 4. Matching and Native Relationships

The term *matching* describes a horizontal relationship between peers. In the MPTN architecture, users must always match. To “match” means to have the same application support and to belong to the same application program family. For example, two sockets programs match or two CPI-C programs match, but a sockets program and a CPI-C program do not.

The term *native* describes a vertical relationship between a user and its corresponding provider of transport services. With respect to a certain user, a transport network that provides the address type and transport characteristics assumed in the user design is native to that user. A transport network that does not provide that address type and those transport characteristics is nonnative. A *native node* is a node with no MPTN capability. For example, a node with SNA application programs running only over SNA transport is a native (SNA) node.

MPTN architecture allows end nodes to access the MPTN network using SPTN protocols only. This is represented graphically in Figure 5.

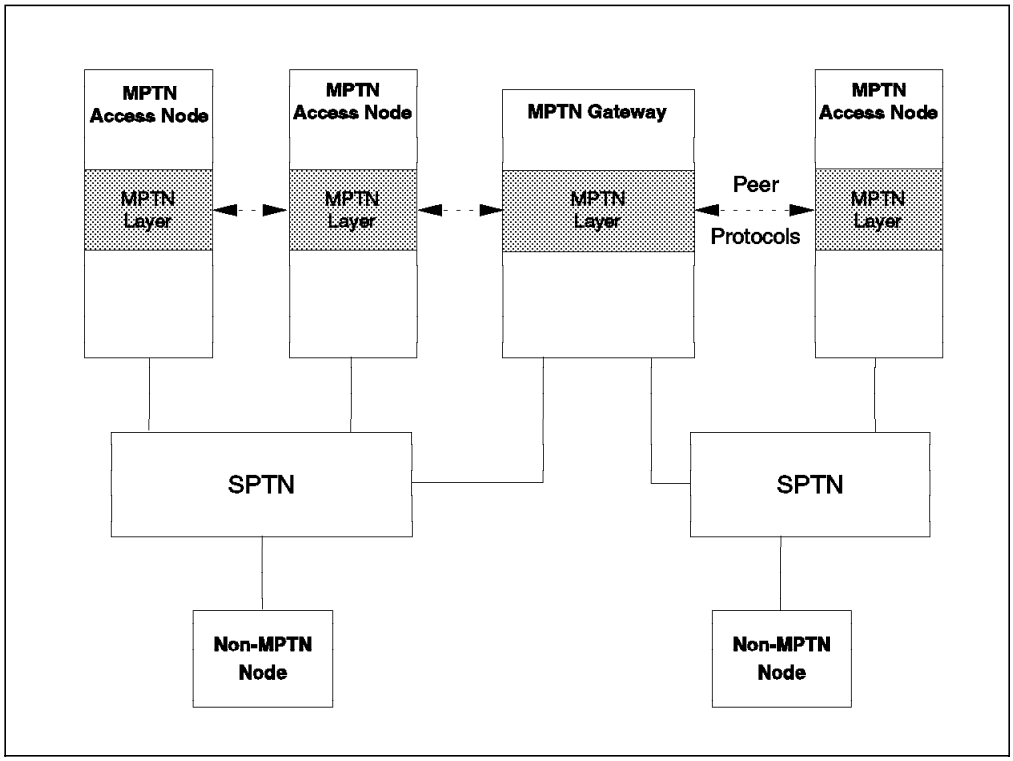


Figure 5. MPTN Network Structure. All end nodes have transport connectivity with each other. The non-MPTN end nodes access the network using their respective SPTN protocols.

The MPTN layer in Figure 5 incorporates a single internet routing protocol that allows traffic to be routed between different SPTNs. MPTN gateways participate in the routing protocols of their respective SPTNs to perform native routing services such as learning and advertising reachability information.

In this manner, isolated subnetworks can be interconnected with no change to the application program by the implementation of the MPTN layer immediately above the transport layer.

1.7 MPTN Architecture

Networking protocols generally provide three types of functions:

- Transport
- Naming and addressing
- Higher level functions

Transport functions are those that provide the basic facility for two partners to communicate, either through connections or in a connectionless manner through datagrams. *Naming and addressing* conventions define how entities are known and how they are to be found. The *higher level functions* include allocation of the connections to users and control of their use. Not all networking protocols support higher level functions. SNA and OSI do; TCP/IP and NetBIOS do not.

MPTN separates transport functions from higher level functions and from the address formats as seen by the users of the protocol. Its goal is to allow *any* higher level protocol using the corresponding address structure to run over *any* transport function. The division of functions between transport and higher level was chosen because the transport level is the highest level at which there are common functions that can be reasonably mapped across protocols. At other levels the number of services is either too large or too small to provide a practical division. This point is explained on 1.3, “Why a Transport Solution?” on page 3.

1.8 MPTN Functions Overview

A *single-protocol transport network (SPTN)* consists of a group of nodes that are physically connected and implement the same transport protocol. These nodes may be connected by protocol-specific gateways, such as IP routers in a TCP/IP network or APPN border nodes or SNI gateway configurations in an SNA network. A *multiprotocol transport network*, as shown in Figure 6 on page 12, is a confederation of SPTNs, each of which has its own transport protocol. The MPTN network appears to its users as a single logical network having the corresponding transport protocol of the user. This single logical appearance is provided by two aspects of MPTN:

- The MPTN access node
- The MPTN gateway

1.8.1 MPTN Access Node

An *MPTN access node* contains the *transport-layer protocol boundary (TLPB)*, which provides a semantic interface so that higher-level protocols or application interfaces written for a particular transport protocol can be transported over another protocol with no apparent change. Such a node can run application programs independent of the underlying transport network and can run application programs on different underlying transport networks.

The applications in the MPTN access node are generally written to an existing application programming interface (API). The API uses the functions of the native communications protocol. When transport-level MPTN functions are accessed, the API will be converted to access these functions instead of the native communication protocol, while keeping the same interface to the application program. For example, the NetBEUI application interface is written to

use the NetBIOS communications protocol. To use another protocol stack below the transport layer, the NetBEUI API must be made to invoke the MPTN functions. After this is done, all programs using NetBEUI on this MPTN access node can communicate via, for example, SNA, TCP/IP, OSI and other protocols, with another NetBEUI application within the MPTN network. All this is possible without the original application program requiring any change!

1.8.2 MPTN Gateway

The *MPTN gateway* connects different single-protocol transport networks. The MPTN gateways provide cross-network *directory* services, and *MPTN segments* located in different SPTNs are concatenated to create an *MPTN connection*.

The MPTN gateway has two functions:

1. It connects non-MPTN environments to the MPTN network.
2. It connects single-protocol transport networks to form the overall MPTN network.

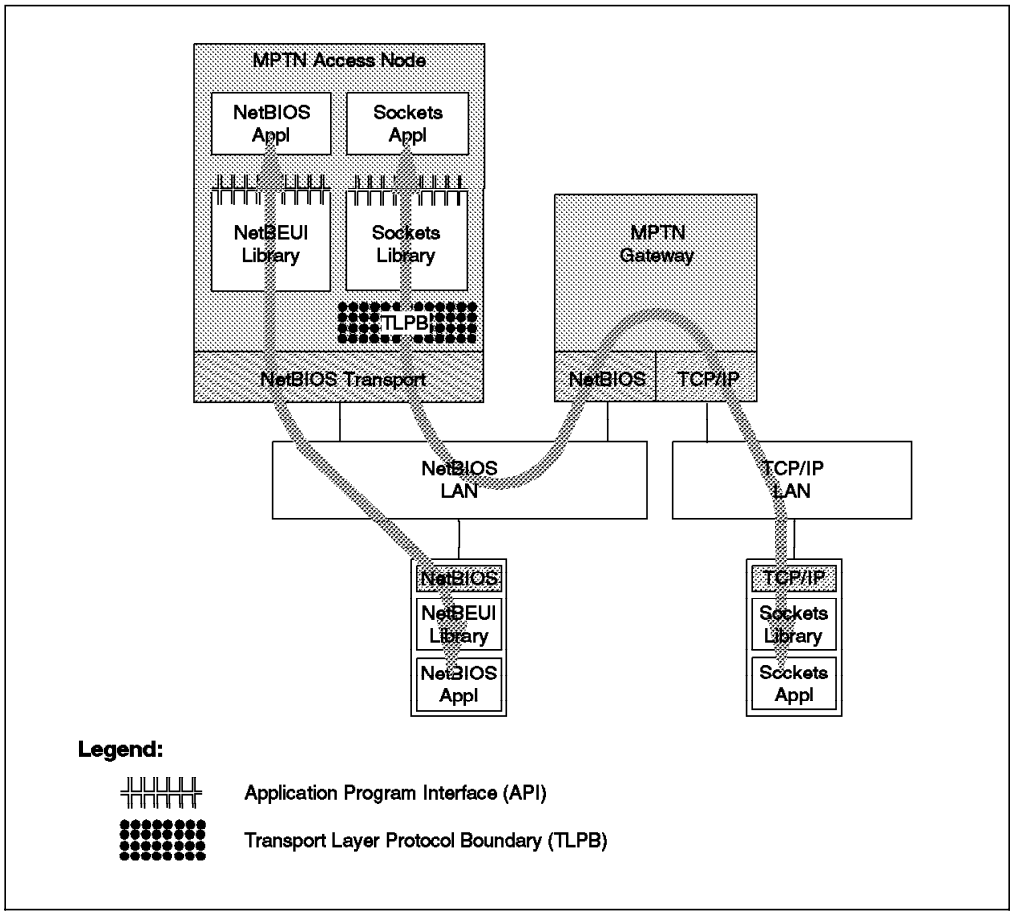


Figure 6. An MPTN Network

Let us look at these from the viewpoint of our sockets application in Figure 6 [1]. MPTN allows existing sockets applications to communicate with each other via a different communication protocol below the transport layer (in an MPTN access node). With function 1 of the gateway we are now also able to interconnect a

[1] The symbols representing the API and the TLPB in this figure will be used throughout this publication for these two interfaces.

sockets application running *natively* over a TCP/IP network to another sockets application running over a non-TCP/IP transport protocol, the IP network being a non-MPTN environment.

Function 2 of the gateway allows two SPTNs (for example, the TCP/IP-based one and the NetBIOS-based one) to be connected so that the sockets applications in both networks can communicate with each other.

Chapter 2. Architecture Overview

As we mentioned in the introductory chapter, MPTN functions fall into two basic categories:

- MPTN provides the framework to allow applications to operate over transport networks for which they were not originally designed.
- MPTN provides both an *access* and *gateway* node function.

As we proceed to describe the architecture of MPTN, we should keep in mind the above two objectives that MPTN is addressing.

2.1 MPTN Methodology

MPTN architecture solves the above requirements by defining a *new* canonical interface to a set of transport services that concatenate connections across multiple networking protocols. Figure 7 illustrates this concept.

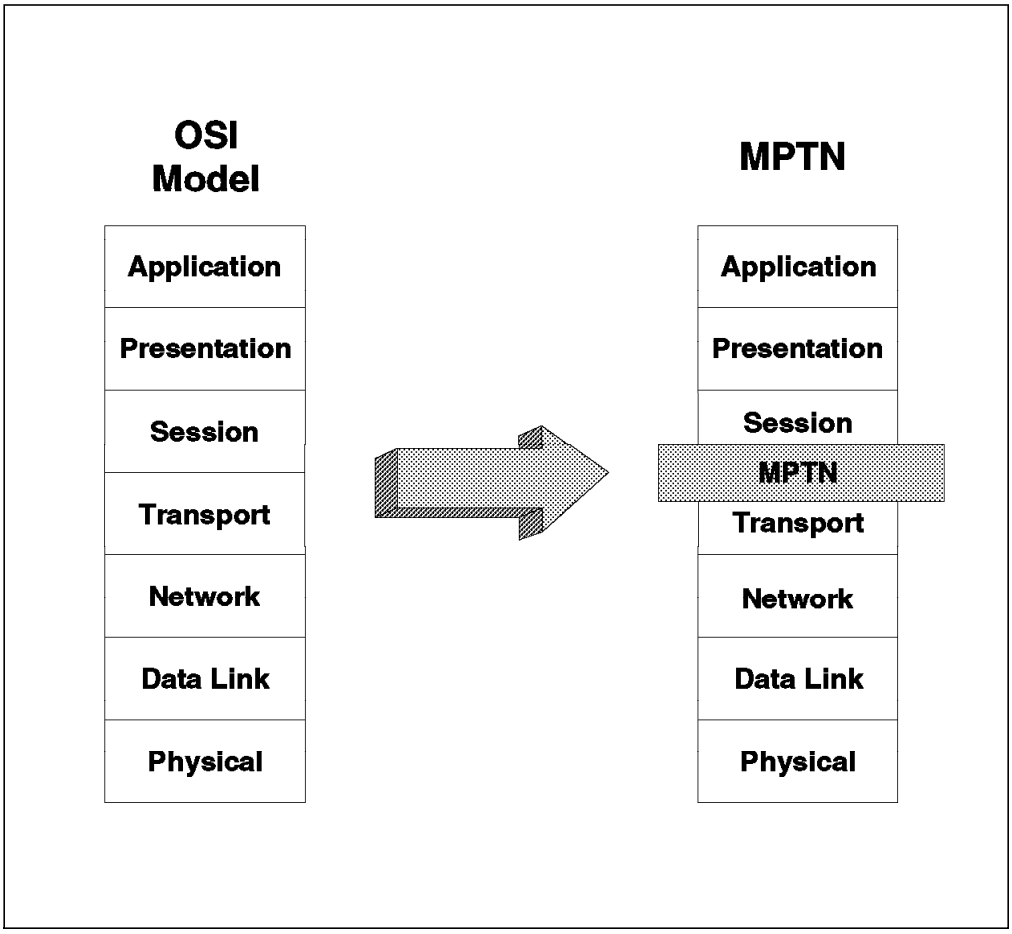


Figure 7. MPTN Interface. This separates the application subsystem from the underlying transport network.

When an application is written for a particular transport service, it may be written so that it makes assumptions about the particular transport service. Thus it may appear to be transport-specific in the services that it uses. For example, applications written to the NetBEUI interface may request a message be

broadcast. In environments where a particular service is not natively supported over the underlying transport network, MPTN provides *compensation*. In essence though, MPTN frees up applications so that they are able to operate over different transport networks.

Another way of thinking about this is that (in the OSI model) functions from the *session layer up* are users of transport services or *transport users*. These services are in turn provided by functions from the *transport layer down*. MPTN defines a boundary interface, called the *transport-layer protocol boundary (TLPB)*, which clearly delineates this distinction between transport user and transport provider. This is shown graphically in Figure 8.

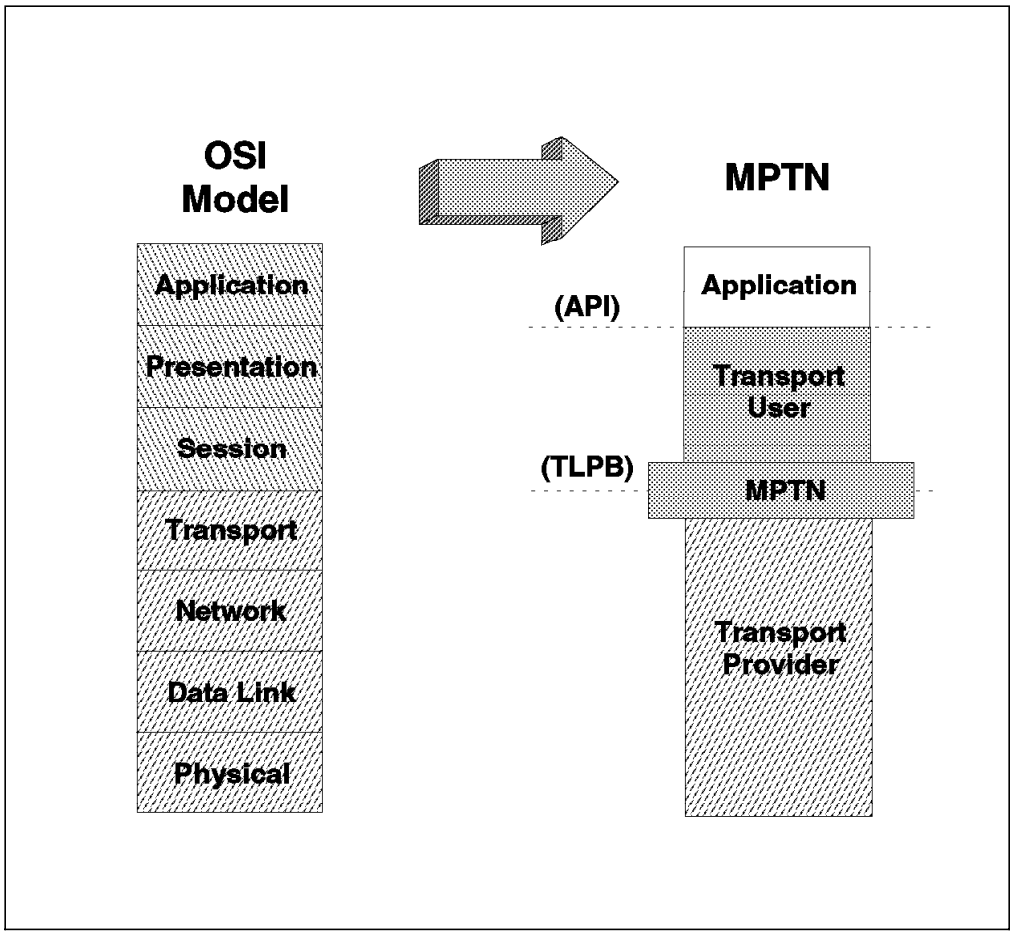


Figure 8. Transport-Layer Protocol Boundary. Transport users are above the TLPB while transport providers are below.

This figure also shows that MPTN defines *transport user* as the functions provided between the API and the TLPB. That is, a transport user is characterized as implementing any higher-layer protocol functions above the transport layer which are provided by the API it supports. A transport user is specific, therefore, to a particular API and the interfaces down to the TLPB. Depending upon the level of function provided at the API, the degree of functionallity within the transport user will vary accordingly. The transport user′s protocol stack has to be modified to interface with the TLPB; transport service requests, native to the transport user′s protocol stack, have to be mapped to TLPB requests. This mapping however will be implemented below the API so that the application program itself does not have to be modified. MPTN

introduces a protocol-specific *syntax mapper* layer below the transport user's API and above the TLPB.

Figure 9 shows a range of applications, transport users, and transport providers within a system (MPTN access node). There can be multiple applications which interface via their associated APIs (being MPTN transport users) to more than one transport protocol.

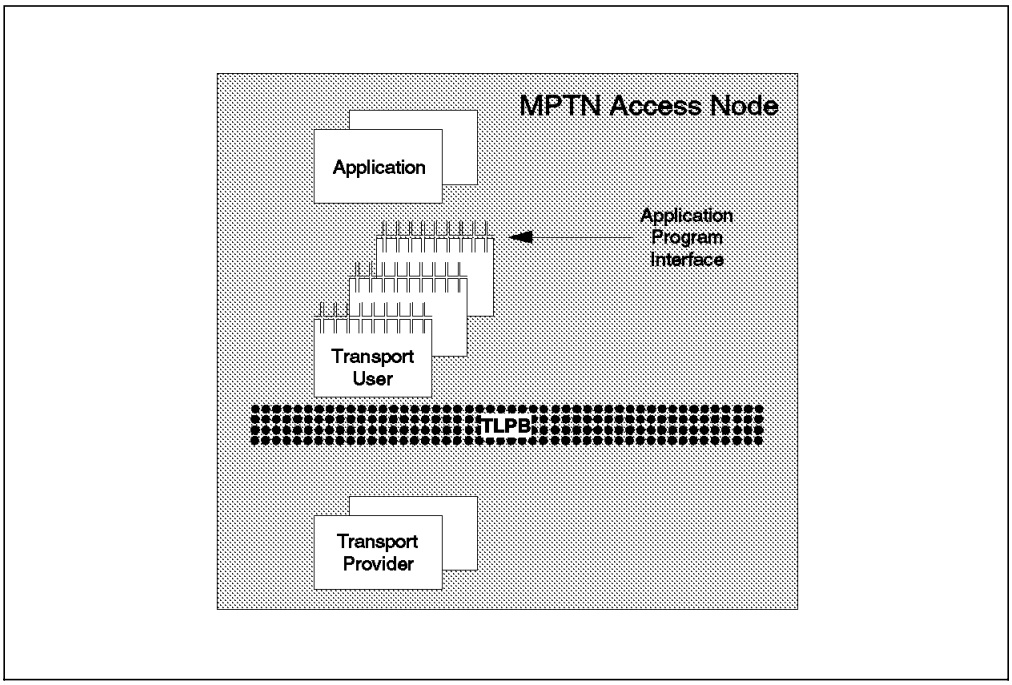


Figure 9. High-Level Structure of an MPTN Access Node

The transport protocols beneath the TLPB consist of the *formats and protocols* of the networking architecture as well as support for any compensatory functions required. The MPTN architecture allows implementations to be designed such that new transport protocols can be added dynamically.

2.2 MPTN Major Components

MPTN functions appear in four types of nodes, termed:

- **MPTN Access Node**

  MPTN access nodes allow an application to run over a *nonnative* transport provider. These nodes may contain an interface that allows programs to be written directly to transport services (for example, the XTI transport interface in OSI or the NetBEUI interface for NetBIOS applications - these transport interfaces run directly above the TLPB as shown in Figure 8 on page 16), or they may contain higher-layer network access services that are themselves users of transport services (for example, APPC which is equivalent to the API interface shown in Figure 8 on page 16), or both.

  Two interoperating MPTN access nodes may be attached to the same transport network (see Figure 10 on page 20), or to networks running different transport protocols (see Figure 12 on page 22) connected through an MPTN gateway.

Through an MPTN gateway, an MPTN access node can interoperate with a native node, that is, one that contains no MPTN function, since the transport user is *native* to the transport provider (see Figure 11 on page 21).

- **MPTN Address Mapping Server Node**

  An address mapping server node is an MPTN node with a special function which provides a general address mapping service to the address mapping client component of other MPTN nodes, access or gateway, connected to the *same* transport provider network. For example, the address mapping server in Figure 10 on page 20 provides the address mapping service to *both* Access Node 1 and Access Node 2. However, Address Mapping Server 1 in Figure 12 on page 22 provides the address mapping service to the address mapping client component of Access Node 1 but *not* to the address mapping client component of Access Node 2, which has to use the address mapping service of Address Mapping Server 2.

  The principles of address mapping are explained further in 2.5, “Address Formats and Mapping” on page 25.

- **MPTN Multicast Server Node**

  A multicast server node is an MPTN node with a special function, the multicast server, which manages multicast and broadcast distribution of datagrams in networks which do *not* provide the service as a natural consequence of their design. For example, a NetBIOS transport network is designed to support multicast distribution of datagrams while an SNA network is not. The multicast server operates in cooperation with the address mapping server to provide a multicast service where this capability does not exist within the services offered by the transport network.

- **MPTN Gateway Node**

  An MPTN gateway connects two transport networks to provide an end-to-end service over both of them for one or more transport user protocols. There are two types of gateways, nonnative-to-nonnative, with respect to the supported transport user protocol, and native-to-nonnative, with respect to the supported transport user protocol.

  In the case of nonnative-to-native one of the transport providers connected to the gateway node is native to the transport user. This type of gateway allows access to an end node that has no MPTN function in it at all. When one side is native, the MPTN protocol (of the nonnative side) *terminates* in the gateway. The MPTN gateway accepts native protocol traffic as requests for MPTN services. For example, an MPTN gateway that is supporting SNA will accept an APPN Locate request and convert it to an MPTN directory query, or it will accept an SNA BIND and convert it to an MPTN connection request.

The different MPTN node types are illustrated in the three configurations of the next section.

2.3 MPTN Configuration Examples

The following pages illustrate the use of the four basic MPTN functions in three sample configurations. The topics are:

- 2.3.1, “Basic MPTN Functions”
- 2.3.2, “Configuration 1 - Single Nonnative Transport Network” on page 20
- 2.3.3, “Configuration 2 - Native-to-Nonnative Transport Networks (Gateway)” on page 20
- 2.3.4, “Configuration 3 - Dual Native Transport Networks (Gateway)” on page 21

2.3.1 Basic MPTN Functions

The MPTN functions identified by the labels MPTN-1 through MPTN-3 in Figure 10 on page 20 through Figure 12 on page 22 are described as follows:

- **Compensation in MPTN Access Nodes and MPTN Gateways (MPTN-1)**

  The MPTN-1 function package represents MPTN compensations required to bridge the gap between the needs of the transport user and the services provided by the underlying transport provider. The same compensation function is required, whether the node is an MPTN access node or an MPTN gateway.

- **MPTN Address Mapping Service (MPTN-2)**

  An MPTN address mapping server maintains a table of mappings from sets of transport-user addresses to one or more transport-provider addresses. The table may be kept on permanent storage or may be kept in memory. In the case where the table is kept on permanent storage, it may be saved over a restart of the node and also possibly shared with other address mapping server nodes. In the case where the table is kept in memory, it is volatile and cannot be shared. The address mapping client component of MPTN access nodes register (transport-user address, transport-provider address) pairs with the address mapping server. Both MPTN access nodes and MPTN gateways request resolution of transport-user addresses to transport-provider addresses.

  The database can be maintained differently by different implementations; only the protocol between the address mapping server and address mapping clients in other MPTN nodes is defined.

- **MPTN Gateway Access to Native Network Routing Mechanisms (MPTN-3)**

  When one side of the MPTN gateway uses a transport network natively, there are some additional MPTN functions involving interaction with the routing and connection establishment protocols of the native network. For example, an MPTN gateway that learns about a set of transport users on the nonnative side needs to feed information into the native routing protocol on the other side so that connection establishment requests or datagrams from the native network will be routed to the gateway. Once these requests reach the MPTN gateway, it then uses basic MPTN gateway functions (MPTN-1 and MPTN-3) to complete the service on the nonnative network.

2.3.2 Configuration 1 - Single Nonnative Transport Network

Figure 10 shows two MPTN access nodes communicating over a single, nonnative transport provider network. Function package MPTN-1 is required for communication across a nonnative network (for a definition of function packages MPTN-1 to MPTN-3 see 2.3.1, “Basic MPTN Functions” on page 19).

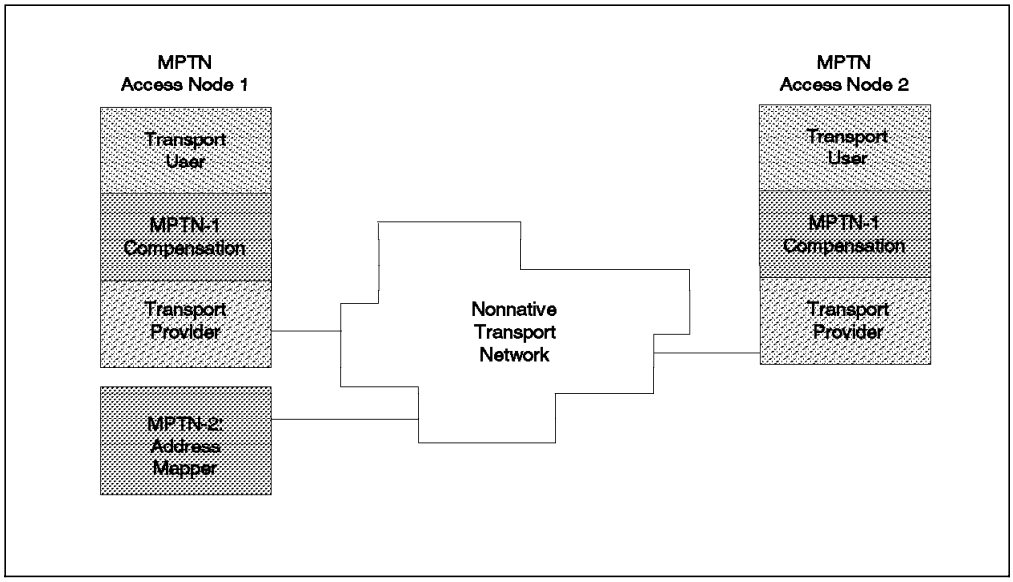


Figure 10. Configuration 1 - Single Nonnative Transport Network

The address mapping server shown in this figure may or may not be needed, depending on the transport provider′s capabilities. For example, a TCP/IP transport provider can provide address mapping via the domain name server (that is, a name is provided to the domain name server which responds by providing the corresponding IP address), while a NetBIOS transport provider has no mechanism for supporting nonnative addresses and hence needs the MPTN address mapping service. The address mapping server employs function package MPTN-2.

2.3.3 Configuration 2 - Native-to-Nonnative Transport Networks (Gateway)

Figure 11 on page 21 shows an application in an MPTN access node communicating through a gateway to a non-MPTN partner in a transport provider network that is native to the transport users. Function package MPTN-3 is required in the MPTN gateway to support communication between an MPTN access node to a native (non-MPTN) node across an MPTN gateway. No MPTN protocol flows at all in the native network.

The gateway interacts with the native routing protocol in the native network.

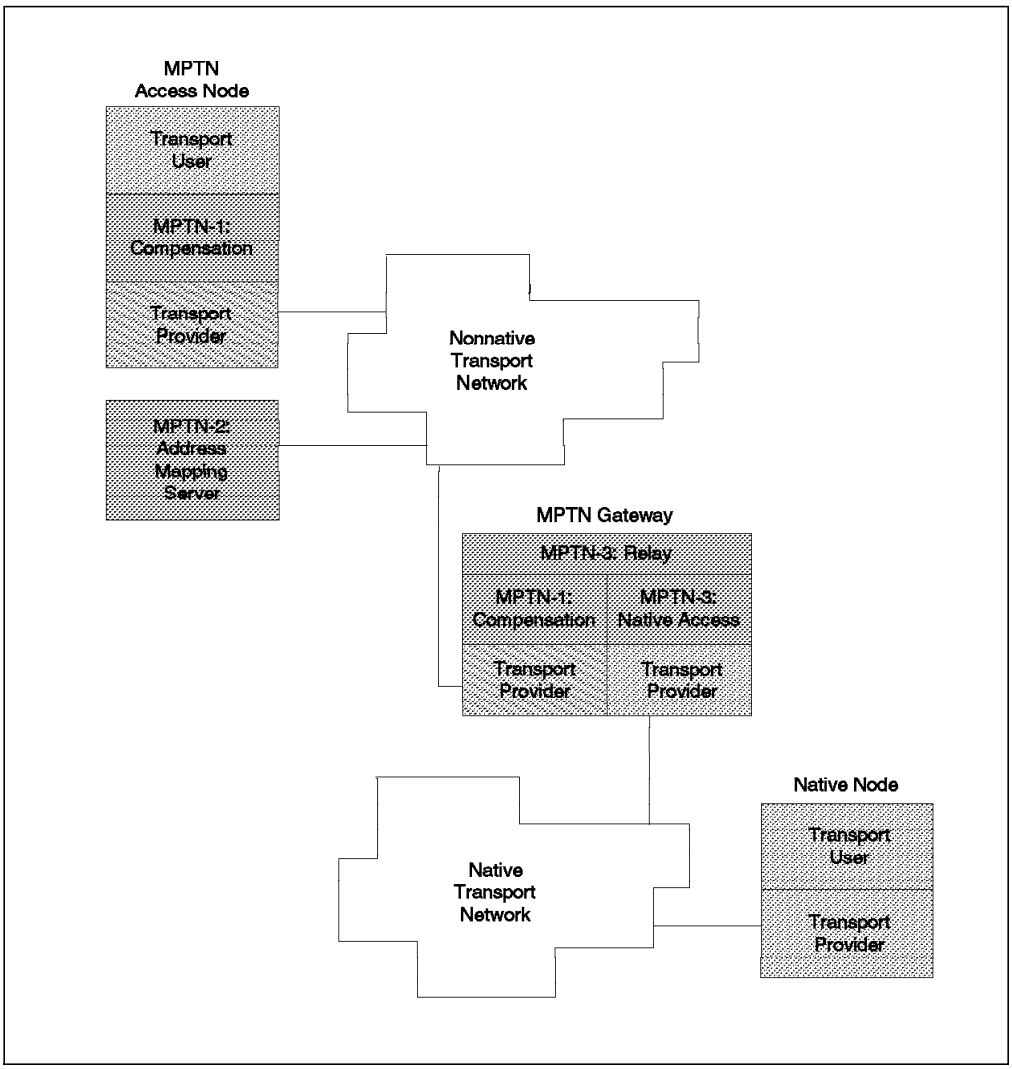


Figure 11. Configuration 2 - Native-to-Nonnative Transport Networks

2.3.4 Configuration 3 - Dual Native Transport Networks (Gateway)

Figure 12 on page 22 shows an application in an non-MPTN node communicating through two MPTN gateway nodes to another non-MPTN node. The non-MPTN nodes use a transport provider network that is native to the transport users. Function package MPTN-3 is required in the MPTN gateway nodes to support communication between another MPTN gateway node and a native (non-MPTN) node across the MPTN gateway. No MPTN protocol flows at all in the native network.

The gateway node interacts with the native routing protocol in the native network.

An address mapping server node has not been shown although it could be present and performing the same function as in configuration 2.

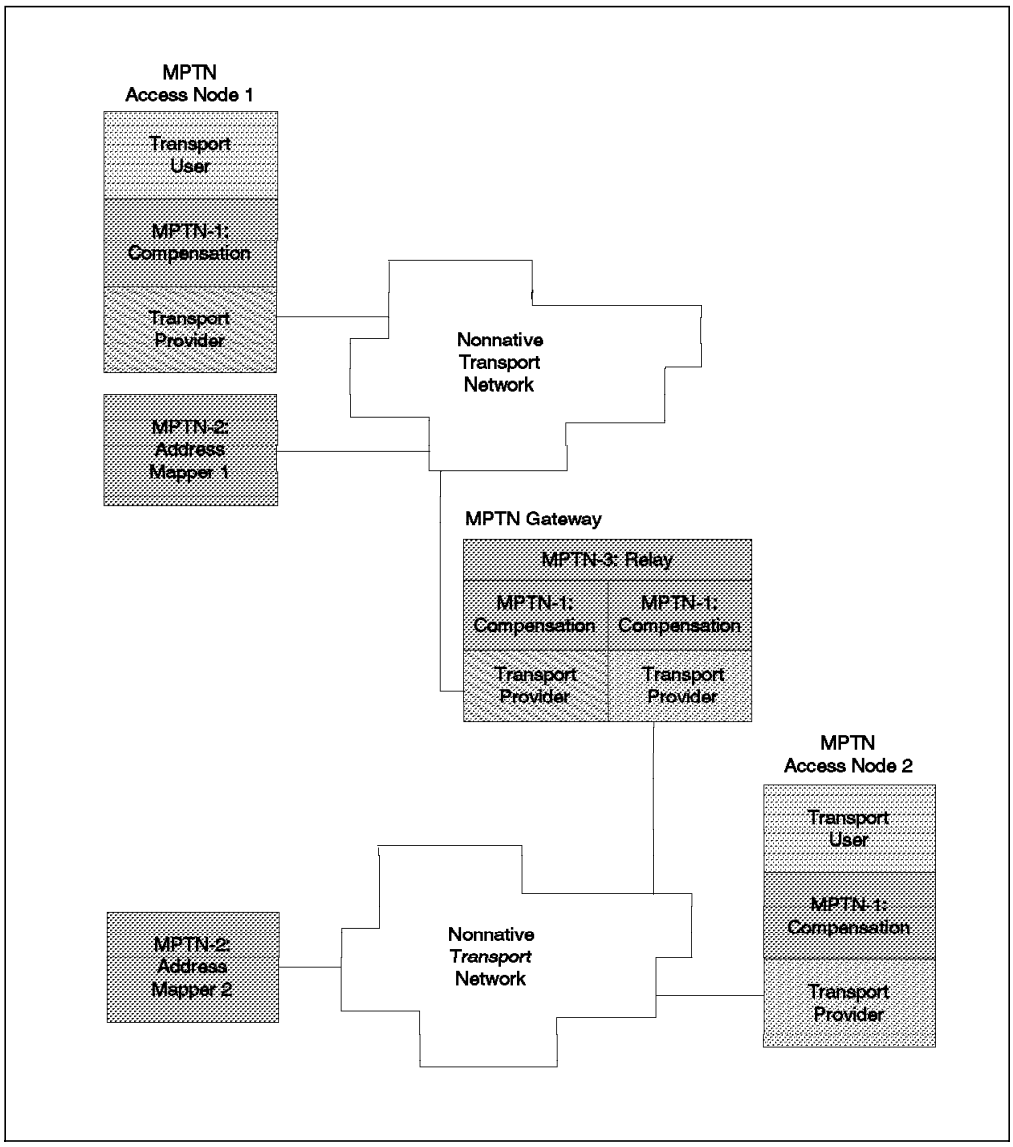


Figure 12. Configuration 3 - Dual Native Transport Networks

2.4 Network Types and Connections

A *single-protocol transport network (SPTN)* consists of a group of nodes that are physically connected *and* implement the *same* network protocol. These nodes may be connected by protocol-specific gateways, such as IP routers in a TCP/IP network.

A *multiprotocol transport network* is a confederation of SPTNs, each of which has its own transport protocol. Figure 13 on page 23 shows an MPTN network made up of two SPTNs.

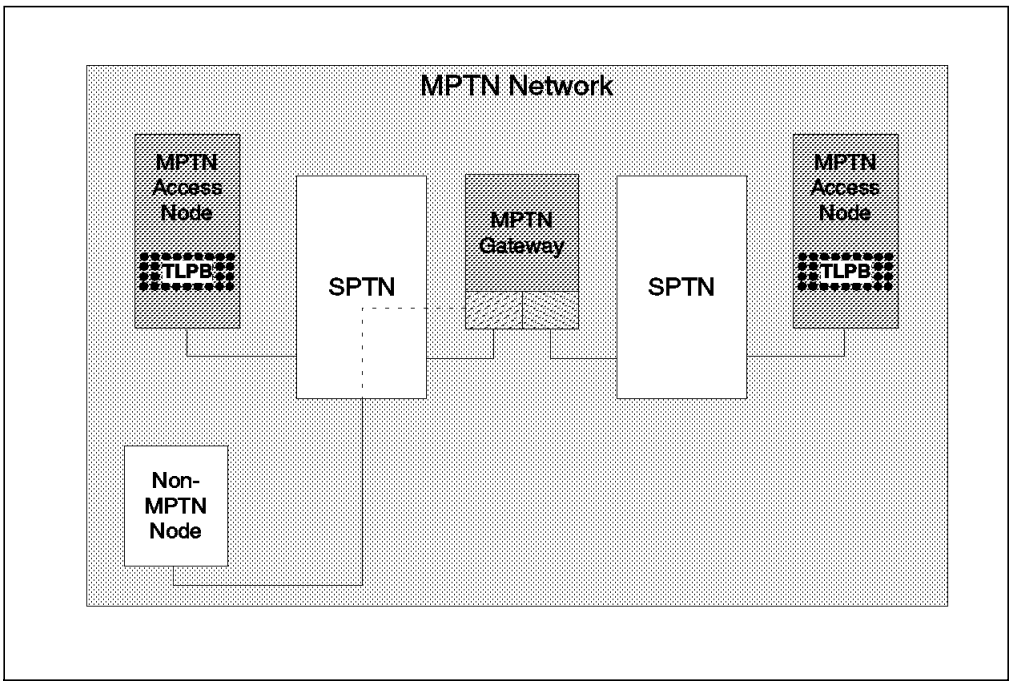


Figure 13. Two SPTNs Connected by a Gateway to Form an MPTN Network

The MPTN network appears to its users as a single logical network having a single transport protocol. The single logical appearance is provided by two aspects of MPTN: the *TLPB* and *MPTN gateways*.

The TLPB provides a semantic boundary so that higher-level protocols or application interfaces written for a particular transport protocol can be transported over another protocol with no apparent change.

MPTN gateways connect the different single-protocol transport networks. The MPTN gateways provide routing services, and they concatenate an *MPTN segment* in one SPTN with an MPTN segment in another SPTN to create an *MPTN connection*. Underlying each MPTN segment is a *protocol-specific connection*.

Each SPTN in an MPTN network may itself be a composite network, as illustrated in Figure 14 on page 24. If this is the case, the MPTN gateway must also function as a *native routing gateway* so that the MPTN gateway can be aware of the transport provider addresses accessible through it. A native routing gateway is a protocol-specific gateway that implements the routing protocol of that SPTN. For example, a TCP/IP network that is a portion of an MPTN network can be a composite network with IP routers (traditionally called gateways in TCP/IP RFCs) connecting multiple IP subnetworks together.

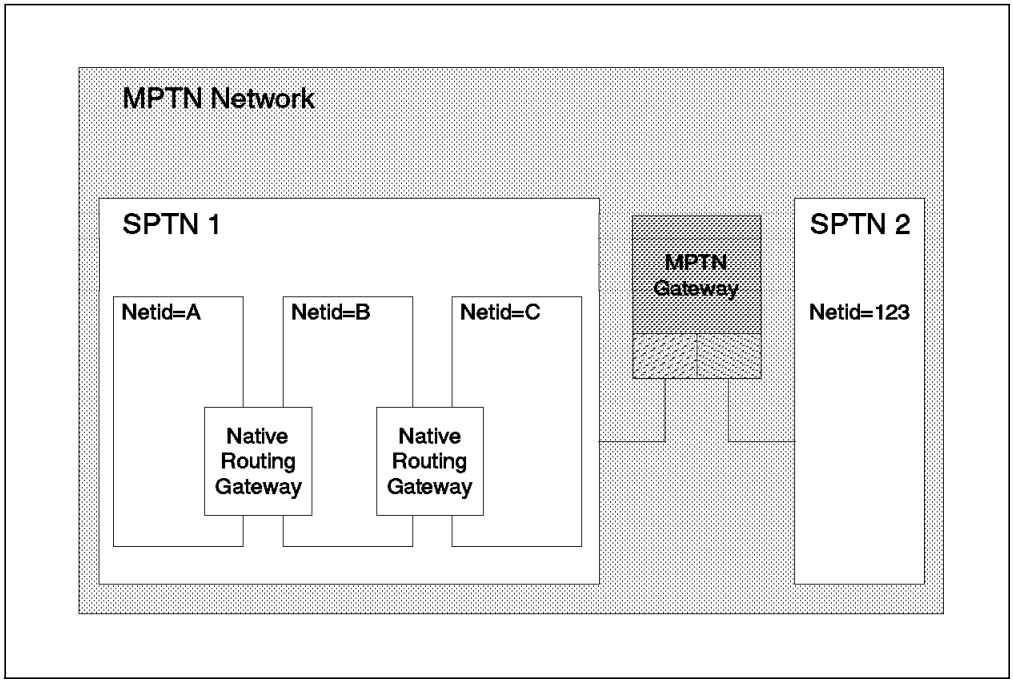


Figure 14. SPTN with Multiple Net IDs

MPTN connections through the MPTN network are formed by concatenating *MPTN segments*. An *MPTN segment* is a connection across a single network between an MPTN or non-MPTN node and a gateway, or, between two gateways. All data received on a given incoming MPTN segment is forwarded on the associated outgoing MPTN segment, as illustrated in Figure 15.

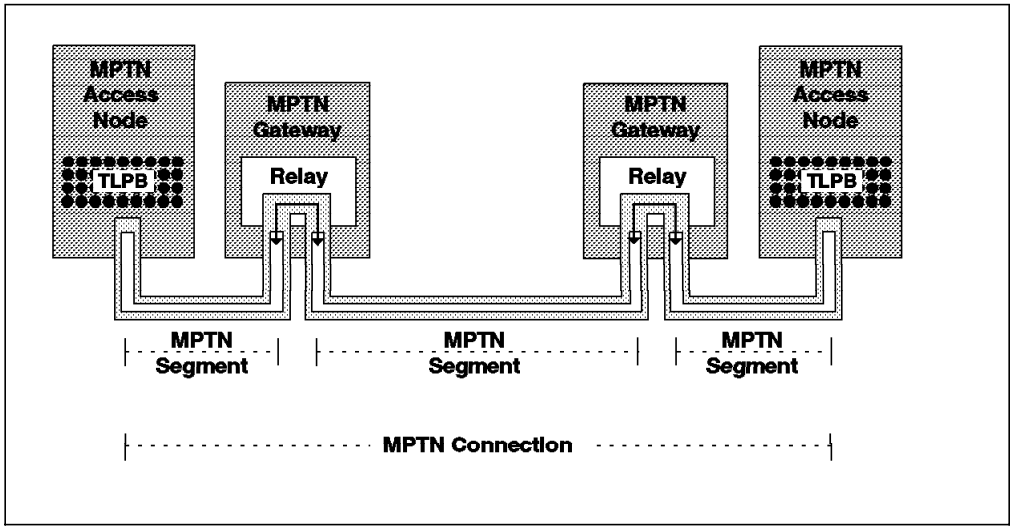


Figure 15. MPTN Connections Supported by Concatenated Segments

2.5 Address Formats and Mapping

Address mapping services are required when transport user and transport provider have different addressing schemes. Since MPTN supports connectivity across heterogeneous networks, it must deal with different address formats of the network protocols it supports. In order to avoid disruption to either transport users or transport providers, MPTN allows a transport user to use its existing address format, while the serving transport provider uses transport addresses that its network expects.

In Figure 16, transport users communicate with each other using transport-user addresses. Because the transport network uses another protocol, transport user addresses are mapped to transport-provider addresses before communication across the transport provider's network is started.

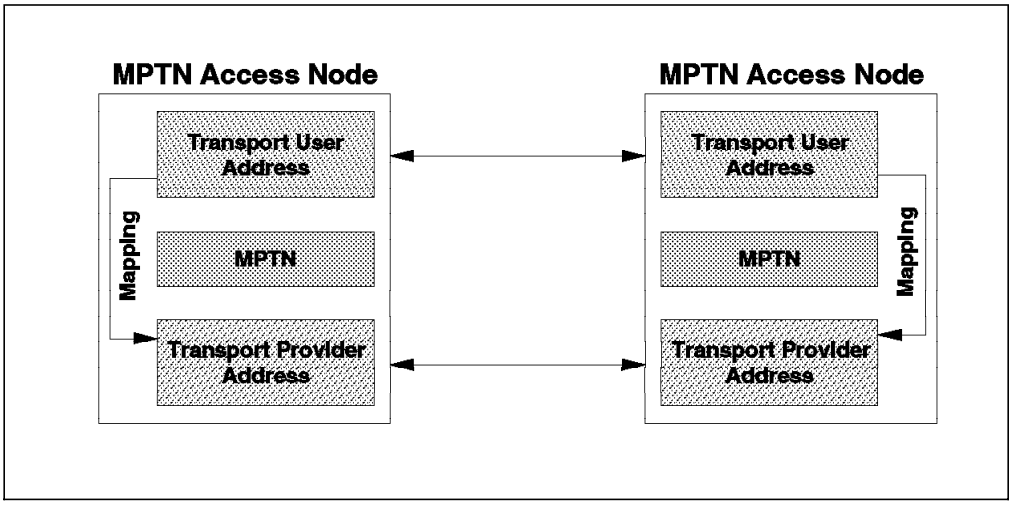


Figure 16. MPTN Address Space Mapping

When a transport user (application) requests services of a nonnative transport provider, MPTN needs to provide the transport user with a network address compatible with the particular transport network being employed. That is, a mapping must be made between the addresses the transport user understands and the addresses the transport provider's network uses. For example, if the transport user is an AF_INET sockets application, then its own address will be of the form *IP address.TCP_Port* or *IP address.UDP_Port*. If this application is run over an APPN network, then its IP host address needs to be *mapped* to an SNA address of the form *Netid.LUname*. That is, it must resolve any naming conflicts prior to returning a mapped name to the transport user. The address mapping service uses the available transport provider's services to communicate with other MPTN nodes, and is thus a transport user itself.

To allow duplicate addresses to be used in different protocol address spaces, MPTN uses an ordered pair (address type, protocol-specific address) to identify its users uniquely in the MPTN network. Such an ordered pair is called an *MPTN-qualified address*.

MPTN provides an address mapping between a transport user's address and a transport provider's address when different addressing schemes are used. For example, these network protocols each use different addressing formats:

- OSI uses network service access point (NSAP)
- IP uses internet addresses
- SNA uses names and addresses
- NetBIOS uses names

The *node address* identifies a node in a network. A node address must be unique so that a node can be clearly identified. Often a small number of node addresses can address the same node.

The *local address*, on the other hand, identifies a specific transport user *within* a node. The local address, which may be generated dynamically, is unique in the node. For example, the pair transport address of TCP/IP consists of an IP address (the node address) and a port number (the local address).

As Table 1 shows, TCP, UDP, and OSI use different forms of node address and local address, but NetBIOS and SNA use the same format for the node address and the local address.

Table 1. Node and Local Address Formats

Network Type	**Node Address**	**(Node Address,Local Address)**
TCP	IP address	IP address.TCP_Port
UDP	IP address	IP address.UDP_Port
OSI	NSAP	NSAP.Tselector
NetBIOS	NetBIOS name	
SNA	Netid.LUname	

For specific protocols, addresses are normally structured as (netid.hostid.local-address), which are represented in MPTN protocol data units as a bit string qualified by the protocol of the type of network. MPTN gateway routing is based on taking a portion of this bit string, starting from the left, and using this as a generic destination. This includes also the possibility to take none of the bit string. This is comparable to the use of a partial or full *wildcard* as found in SNA APPN routing.

MPTN has a component that maps between the transport addresses used by transport users and those understood by nonnative transport providers. MPTN performs the following two major address mapping functions in order to map a transport-user address to a transport-provider address *dynamically*:

- A transport user registers an address, making itself accessible to other transport users.
- MPTN locates a partner, given a transport-user address. This allows a user to set up a nonnative connection with, or to send a nonnative datagram to, another transport user.

2.5.1 Three Approaches to Address Mapping

MPTN uses three approaches to address mapping:

1. **Algorithmic Mapping**

   With algorithmic address mapping, an algorithm is applied to the transport provider's address A which results in a corresponding address B which is used across the transport provider's network. This is illustrated in Figure 17.

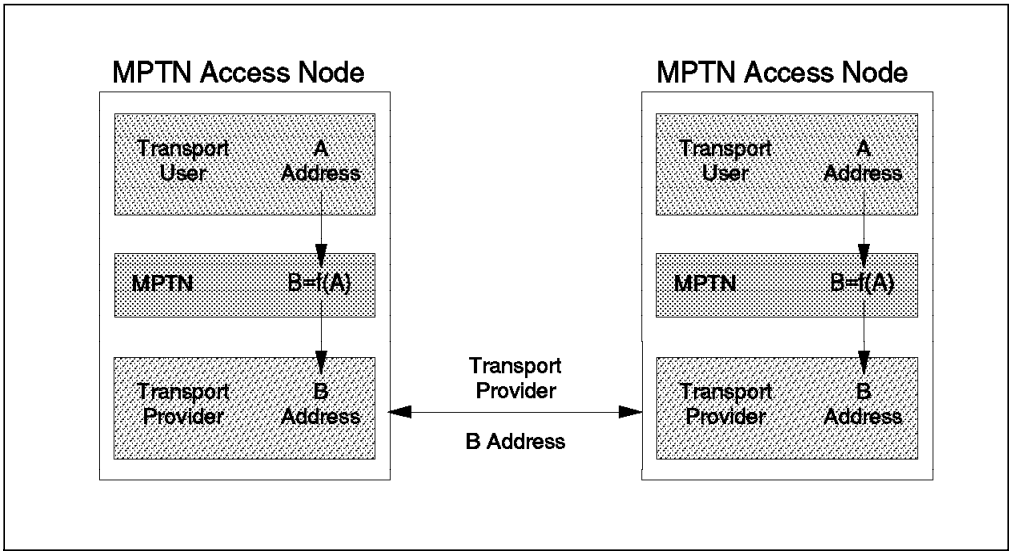


Figure 17. Address Mapping: Algorithmic

   The new transport-provider address may be registered in the existing local transport provider's directory. This algorithmic method is appropriate when the transport user's address space can be mapped into the transport provider's address space.

   An example is mapping an internet address to an SNA name when a TCP/IP application wants to communicate across an SNA network. See 7.2.7.1, "Mapping Internet Addresses to LU Names" on page 109 for a description of the algorithm implemented in AnyNet/MVS for sockets over SNA.

2. **Extended Native Directory**

   A second alternative is to enhance protocol-specific directories to handle transport addresses of various formats. The extended native directory of the transport provider holds the mapping from transport-user address and protocol identifier to the transport-provider address. With this alternative, all transport-user addresses and their protocol identifiers are registered in the transport provider's native directory.

   For example, in Figure 18 on page 28 when transport user A is registered in the transport provider's directory, both a protocol identifier and the user address A are registered along with the association with transport-provider address B. When another transport user requests a connection or datagram delivery to transport user A, a locate request is sent to the transport provider's native directory to find the transport-provider address of the destination.

This method is appropriate when the transport provider's directory supports the registration of address types different from the transport provider's native type.

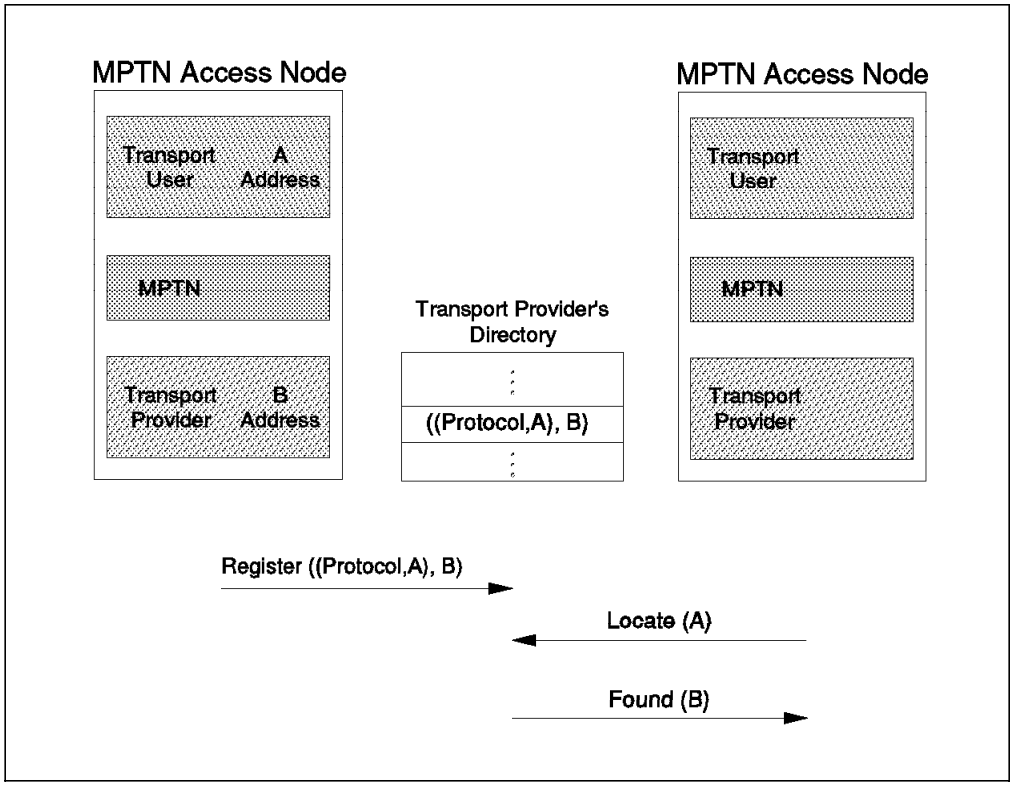


Figure 18. Address Mapping: Extended Native Directory

For our example, because the TCP domain name server can support SNA names, the SNA transport-user address NETA.LU1 is registered as LU1.NETA.SNA.IBM.COM in the domain name server.

3. **Address Mapping Service**

   The address mapping service can be used, especially when neither of the first two alternatives would be appropriate or possible. The address mapping server maintains a table, in memory or on permanent storage, which holds the transport-user to transport-provider address mappings. The table is filled dynamically as transport users connect to the MPTN network and their addresses are registered.

   Figure 19 on page 29 shows how the basic address mapping server functions are performed:

   - An MPTN access node registers a transport-user address and its associated transport-provider addresses with the address mapping server. When the MPTN access node supports more than one transport provider, the transport user can be reached through all transport providers supported by its MPTN access node and, consequently, has a transport-provider address associated with its transport-user address for every transport provider supported by its node.
   - When another transport user requests a connection or sends a datagram to transport user A, its MPTN access node asks the address mapping server for a transport-provider address to be used to reach A. The address mapping server finds the transport-user address in the database

and returns all transport-provider addresses that were registered. If multiple transport-provider addresses are returned, MPTN functions then have to select the transport provider to reach “A” and use the associated transport-provider address.

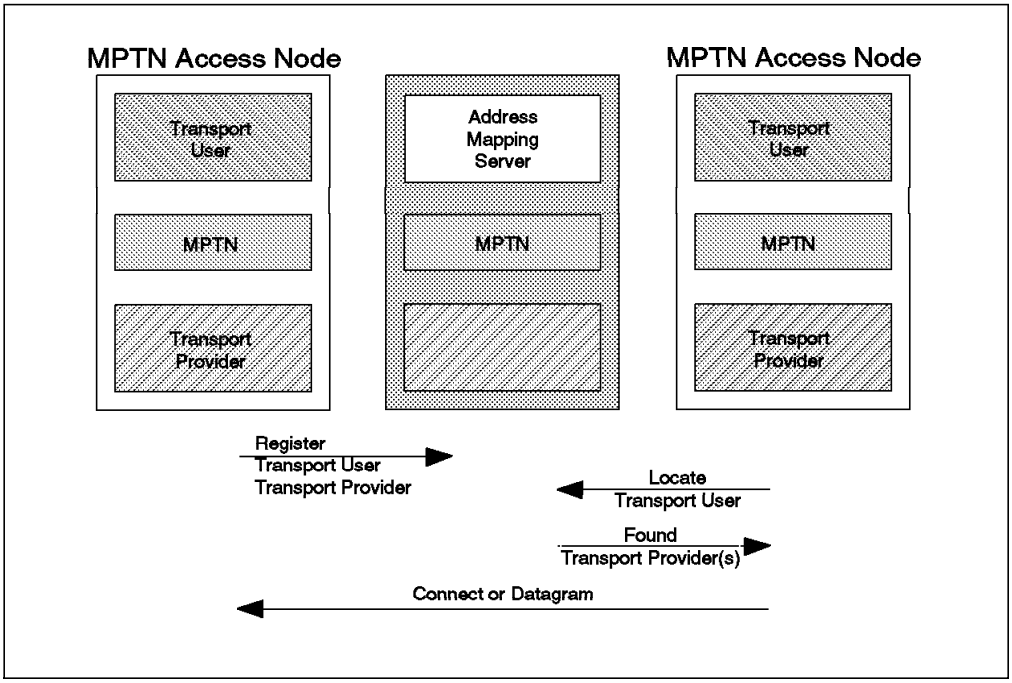


Figure 19. Address Mapping: Address Mapping Server

An example is mapping NetBIOS 16-byte names to SNA names.

Table 2. Advantages and Disadvantages of Each Method

Algorithmic	**Extended Native Directory**	**MPTN Address Mapping Service**
present in all Access Nodes	central on SPTN	Central on SPTNs and, maybe, MPTN
specific to transport user/provider protocols	specific to transport user/provider protocols	not protocol specific
requires conversion from smaller to larger address space	a format conversion will be required	no restrictions
may allow dynamic, effectively, registration	may allow dynamic registration	allows dynamic registration
no network traffic	involves network traffic	involves network traffic

2.6 Connection Establishment

An *MPTN connection* is a connection that can traverse one or more networks that support different transport protocols. Figure 15 on page 24 shows how the MPTN gateway concatenates MPTN segments to form an MPTN connection.

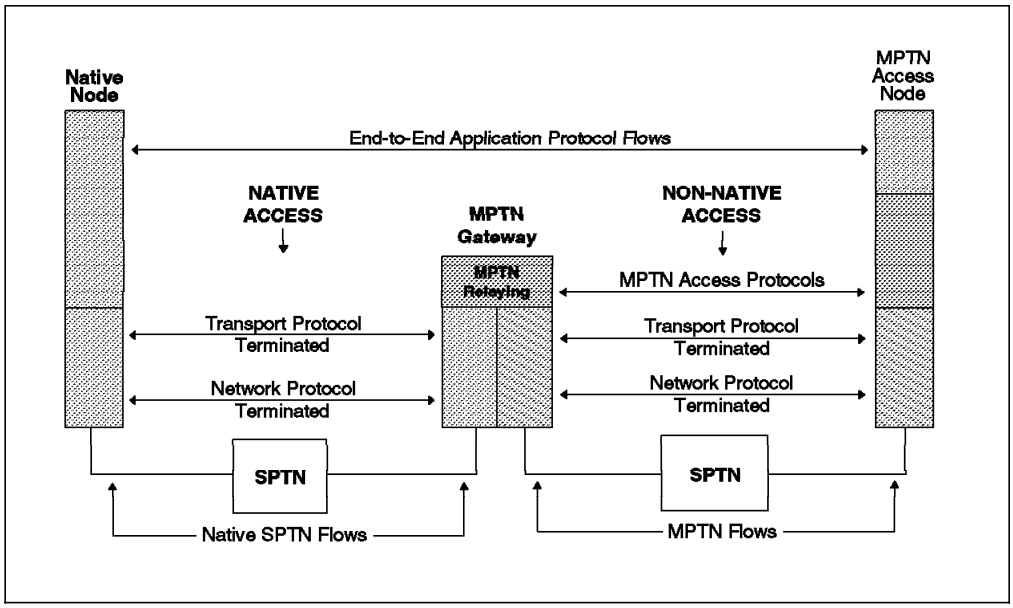


Figure 20. MPTN Connections - Protocols Involved and Areas of Influence

An MPTN connection is made up of the following three types of MPTN segments:

- Source-to-gateway MPTN segment
- Zero or more gateway-to-gateway MPTN segments
- Gateway-to-destination MPTN segment

In the following sections, two of these three segments are discussed and Figure 20 is provided for illustration.

- **Source-to-Gateway MPTN Segment**

  There are two ways the connection from source node to an MPTN gateway can be established:

  1. Native protocols are used where possible to support non-MPTN destinations. When native protocols are used, connection data is sent natively (for example, session parameters in the BIND command), and transport characteristics are inferred for the transport provider type.
  2. When native protocols cannot be used, the MPTN connection setup protocol is used. Transport user information is exchanged in the MPTN connection setup sequences, which include the transport-user addresses, optional connection data, and the transport characteristics selected by the transport user.

- **Gateway-to-Destination MPTN Segment**

  When the MPTN connection arrives at the destination MPTN gateway, the gateway will first determine if the target can be reached using native protocols, that is, over a native MPTN segment. It determines this by seeing if the target node address is supported natively and if the connection

characteristics match those of the transport provider. If they do, the gateway will use a native connection protocol to establish a native MPTN segment. If connection data is used in the native protocol, the connection data that has been passed along in the MPTN connection formats is used in the native connection flows.

If the target node address is not supported natively or if the connection characteristics do not match those of the native transport provider, MPTN connection protocol will be used to establish a nonnative MPTN segment. This must be done to pass protocol-specific connection information across the unlike target transport network.

2.7 Gateway Relaying for Connections

Relaying is the function of transporting the data from the incoming transport provider to the outgoing transport provider in the MPTN gateway. Since the higher-level protocols of the source and destination transport users must match, there will be no conversion of data following the MPTN header. MPTN uses *header deletion* or *header insertion* which involves removing the current transport protocol headers and, possibly, the MPTN headers, and adding transport protocol headers and, possibly, the MPTN headers, for the next MPTN segment.

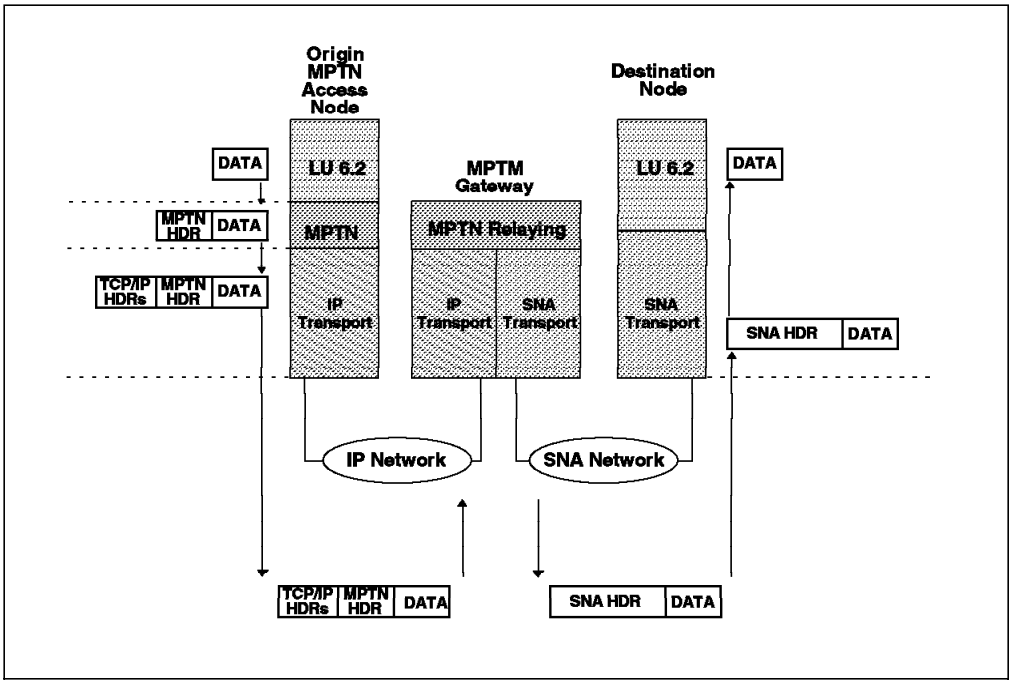


Figure 21. MPTN and Transport Header Levels. The MPTN gateway uses header substitution.

An example for this is illustrated in Figure 21. As the MPTN connection traverses more than one type of transport network, an MPTN header *may* be appended to the user data to indicate any compensation that would be required. In our sample diagram, the MPTN header is required as SNA (which supports a record-oriented data flow) is being sent across an IP network (which supports a stream-oriented data flow). As the user data leaves the *origin* MPTN access node, the transport-specific header is also appended (in this case an IP header). The MPTN gateway relay function receives the frame on the IP-network side,

removes the TCP/IP and MPTN headers and substitutes an SNA transport header in its place. If the gateway is passing from a native to a nonnative network with respect to the transport user, MPTN headers will be added to handle the *compensations* required. If the gateway is passing from a nonnative to a native network with respect to the transport user, MPTN headers will be removed. This is the case illustrated in Figure 21 on page 31. The addition or removal of MPTN headers in MPTN gateway nodes closely matches the requirements for an MPTN access node acting as an origin node or a destination node. One difference is that, if the incoming message is segmented, an MPTN gateway will *not* reassemble it. Also, if an outgoing message is too big, the gateway segments it.

For connections, the protocol-specific flow-control protocols control the flow of data within each network. The MPTN gateway participates in the network flow-control protocols of both networks being concatenated. By judicious allocation of buffers and control over buffer sharing, buffer management in the gateway is able to control the data flow between the concatenated MPTN segments.

2.8 Native versus Nonnative Transmission

Let us now take a specific example to illustrate how the data is formatted when it is routed within an MPTN access node, depending upon whether the transport provider is native or nonnative to the transport user application.

For our example we will assume we have an SNA APPC application communicating over the native SNA transport network and the nonnative TCP/IP transport network.

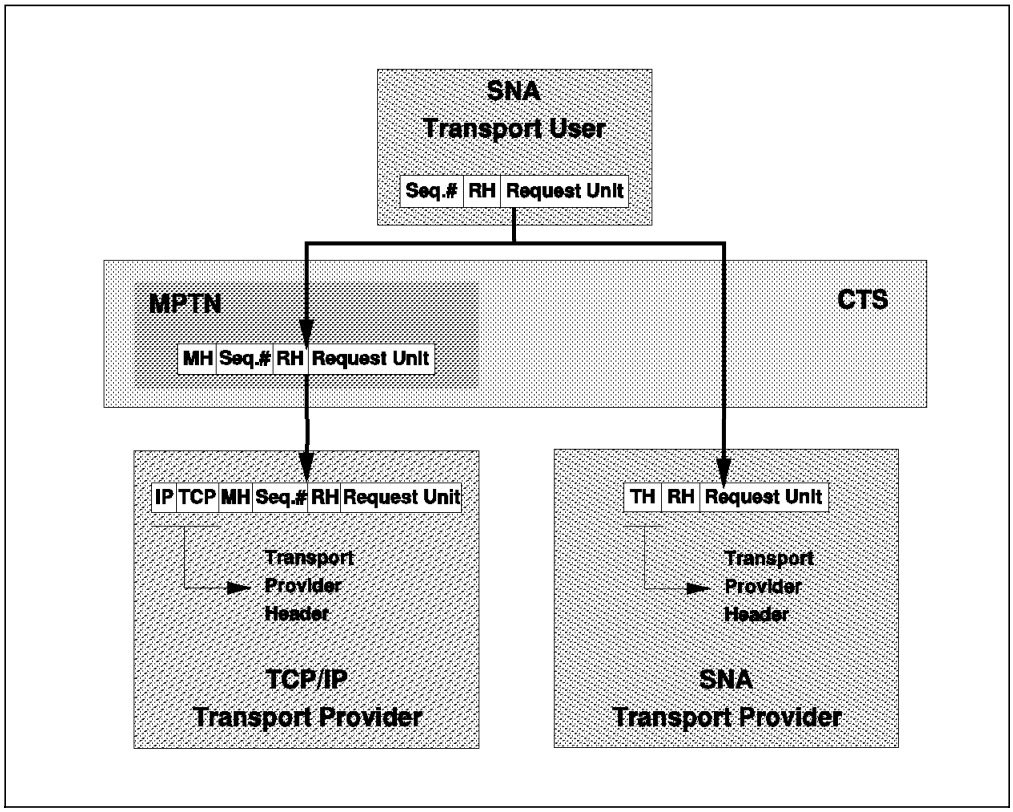


Figure 22. Headers for Native SNA versus Nonnative TCP/IP

In Figure 22 we can see that when the transport network is *native* to the transport user, SNA in our example, then the data may (RH + RU) bypass any MPTN function; no headers are appended to the data prior to being passed to the SNA transport provider which appends the SNA transmission header.

However, when the transport provider is *nonnative* to the transport user, TCP/IP in our example, then the data together with the sequence number, which is normally included in the SNA transmission header, is processed by the MPTN function which adds an *MPTN header* prior to passing the data to TCP/IP for transport. The TCP/IP transport provider then appends the TCP and IP headers to the MPTN-formatted data packet.

2.9 MPTN Examples and Solutions

In the following examples we will show the benefits of MPTN in multiprotocol environments.

2.9.1 Clients and Servers on Different Protocols

Problem:

There is a distributed LAN supporting applications based on NetBIOS. One potential client workstation for the application is *not* attached to the distributed LANs but does have access over an SNA wide-area network.

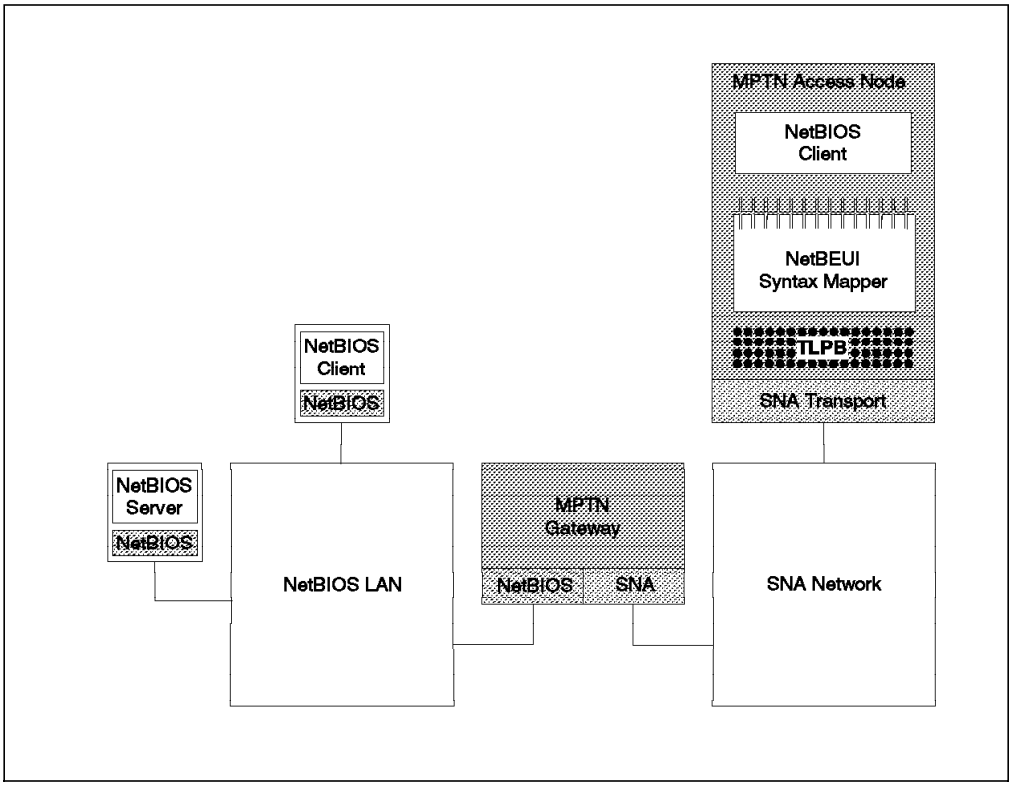


Figure 23. MPTN Solution - Clients and Servers on Different Protocols

Non-MPTN Solutions:

Install a separate application on the isolated client workstation and the server which allows the same client/server function to be performed over SNA.

MPTN Solution:

As shown in Figure 23 on page 33, install an MPTN gateway between the NetBIOS LAN and the corporate SNA network and upgrade the isolated client to an MPTN access node. Consequently, neither the client nor the server applications need be changed in any way and, indeed, there are no changes whatsoever needed in the server node. With MPTN, the LAN is running NetBIOS and is *not* changed. In order to communicate with the server on the NetBIOS LAN, the isolated client application uses the NetBIOS interface and a syntax mapper that maps the NetBEUI calls to the TLPB.

2.9.1.1 Merging Two Unlike Networks

Problem:

One company acquires a second company whose network runs a different protocol. While most business processing need not be merged, a common set of applications is required for financial and personnel processes.

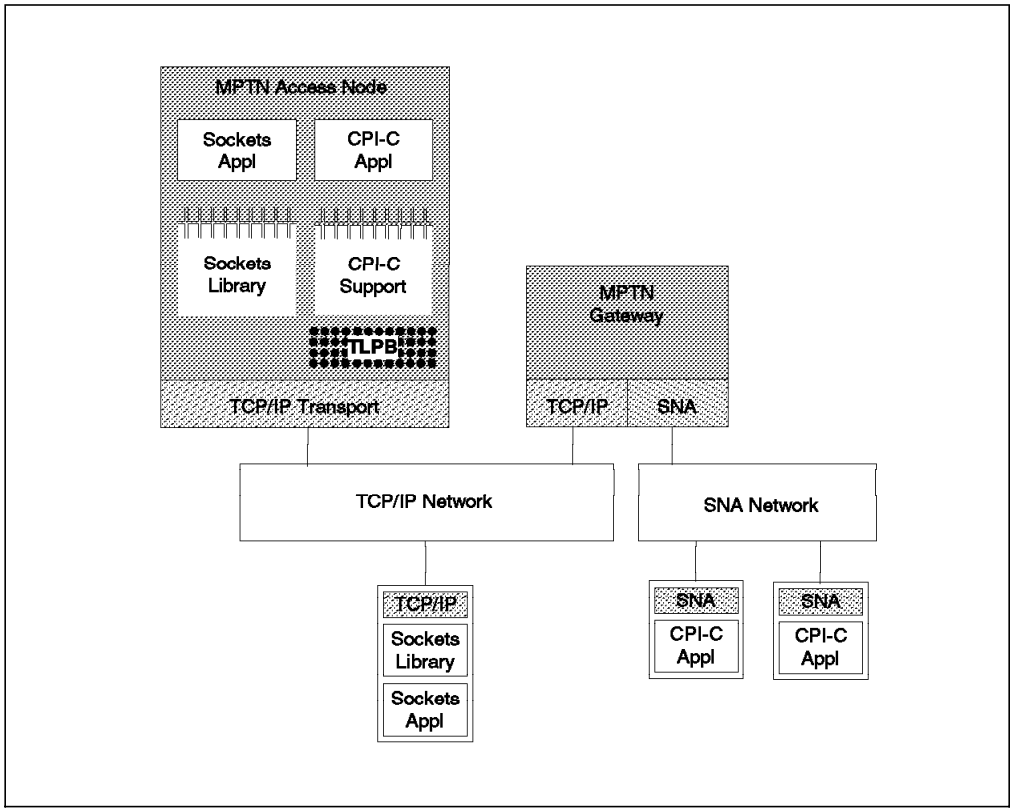


Figure 24. MPTN Solution - Merging Two Unlike Networks

Non-MPTN Solutions:

Install a parallel network in the acquired company and run double protocols where needed, or, rewrite the applications to run on the acquired networks and install application gateways between the two networks.

MPTN Solution:

Install an MPTN gateway between the two networks and an MPTN access node in one of the networks. The two businesses can continue to run all the programs they run today as well as run the parent company's network. In Figure 24, the

company owning the SNA network acquired the the company owning the TCP/IP network. The personnel and financial applications written to the CPI-C interface now can run on the MPTN access node. Of course, as with all program movement from one operating system to another, the cost of porting systems is heavily dependent on the similarities of the operating systems and the extent to which the operating system facilities are used.

These CPI-C applications can communicate with partner applications on the SNA network; the sockets applications running in that node can continue communicating with other sockets applications running on non-MPTN nodes in the TCP/IP network.

2.9.1.2 Departmental LAN Communication

Problem:

Different departments have installed their own LANs with their own protocols and now need to work together on a project.

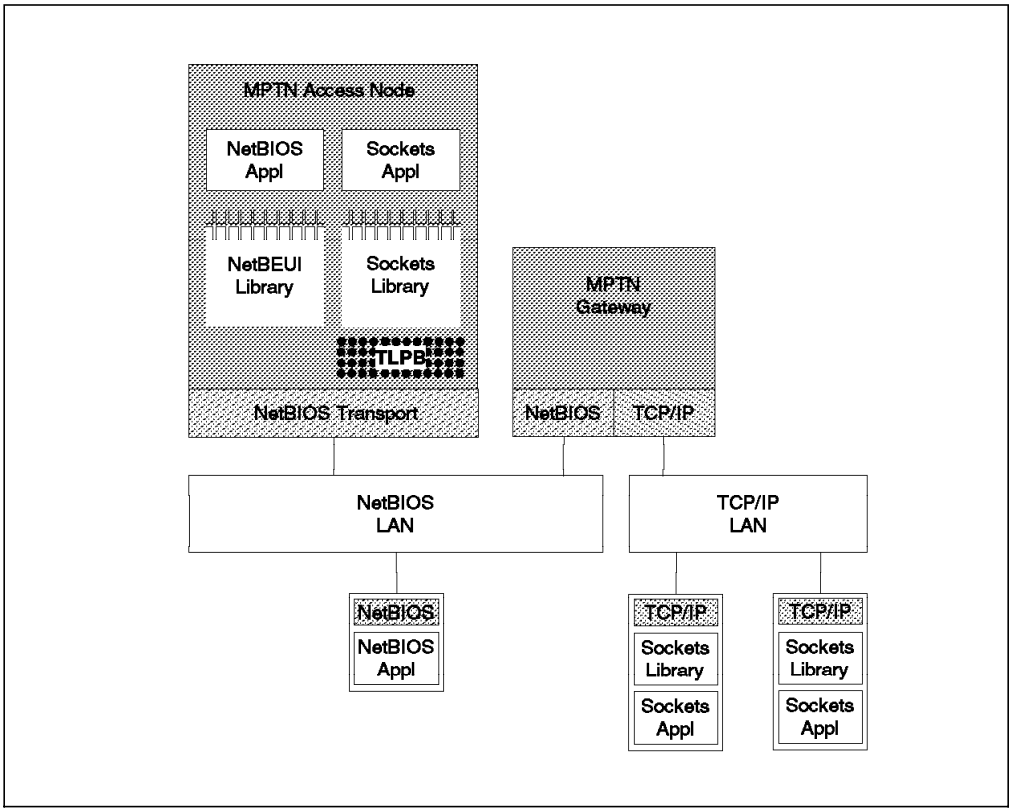


Figure 25. MPTN Solution - Departmental LAN Communication

Non-MPTN Solutions:

Run a second protocol on one of the LANs and install a second complete protocol stack on the workstations that need to communicate with the other LAN.

MPTN Solution:

As shown in Figure 25, place an MPTN gateway between these two departmental LANs, and install MPTN access nodes in the NetBIOS organization to allow sockets applications to run on those machines. MPTN access nodes must contain a sockets library implementation that accesses the TLPB, but it

does not have to contain the TCP/IP code. In this configuration, the sockets applications can run on both LANs, but the NetBIOS applications are limited to running on the NetBIOS LAN.

2.9.1.3 Single-Protocol Workstation

Problem:

Users must access applications on different networks; consequently their workstations have been configured to support multiple protocols. This is costly to the company in terms of software license fees, memory for each workstation, and time spent configuring the workstations.

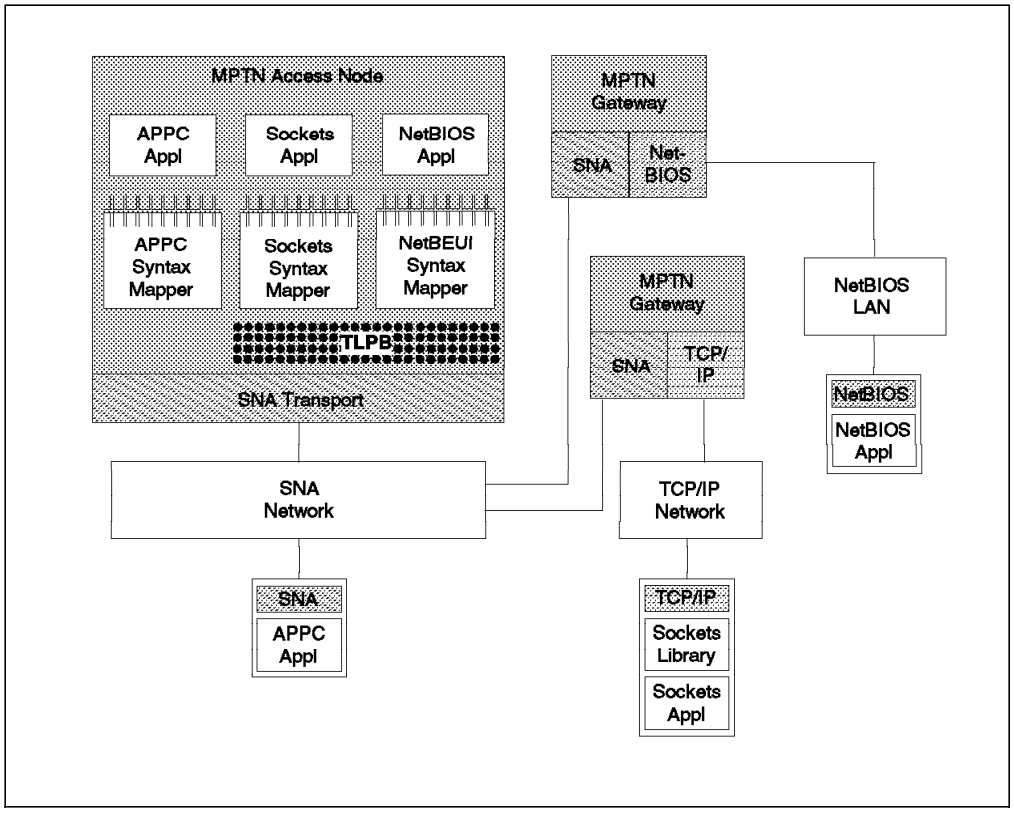


Figure 26. MPTN Solution - Single-Protocol Workstation

Non-MPTN Solutions:

None.

MPTN Solution:

To eliminate multiple protocols, convert each workstation to become an MPTN access node to allow applications that expect different protocols to run in that workstation. In addition, add one or more MPTN gateways at appropriate places to allow LANs to communicate with native networks and non-MPTN nodes. A possible network configuration is shown in Figure 26.

2.10 Summary

MPTN is an open architecture for multiprotocol networking. It allows end users to communicate using whatever transport protocol is available across heterogeneous transport networks. The MPTN architecture makes heterogeneous internetworking possible. With MPTN, customers can choose an API on which to write their applications based solely on the functions the API supports rather than on the protocol installed. MPTN provides a means to support many different APIs over multiple transports at a reduced cost.

Chapter 3. MPTN Transport Services

In a communications protocol stack, the transport layer provides the services to transport data in a distributed processor environment. A variety of communications protocols have been developed to do this and sometimes additional functions are provided with the layers above the transport layer. The transport layer oversees communications across multiple underlying transport facilities. Basic functions provided by this layer are:

- Making an address known to the communications community
- Looking for a communications partner
- Starting communications with the partner
- Exchanging data
- Ending communications
- Removing an address from the communications community

Nearly all communications protocols provide these functions in a similar way, even if the details differ. This chapter discusses the set of transport services which are defined in the transport-layer protocol boundary (TLPB) of the MPTN architecture.

3.1 Services of the Transport-Layer Protocol Boundary (TLPB)

The TLPB defines a set of transport services that are provided to transport users across an MPTN network. It is a *protocol boundary* not an *interface*, which means that an MPTN implementation must provide the same functions, architecturally speaking must use the same *semantics*, but the implementation specifics are open. This means there is no fixed *syntax*.

The set of transport services that is defined in the TLPB was originally derived from the services provided by four common transport providers: NetBIOS, OSI layer 4, SNA, and TCP/IP. A later exercise in mapping the Novell** SPX/IPX in this context confirmed that the proposed TLPB is general enough for a broad class of transport protocols, and the inclusion of new protocols only requires minor additions, if any. That is, MPTN functions do not provide only a subset of functions common to all these protocols. They cover those services considered to be generally useful and necessary for transport. But no single transport provider protocol supports all of these TLPB functions. For example, SNA, NetBIOS, and OSI support a record model while TCP/IP supports a stream model. To implement the record and the stream model, compensation for missing functions must be performed by MPTN functions.

An overview of compensations required between the different protocol stacks is given in Figure 27 on page 40. As you can see from this example, NetBIOS supports a record data model and multicasting, but does not support user expedited data and a stream data model; SNA supports a record data model and user expedited data, but no multicasting. This means that for a NetBIOS over SNA combination, compensation is required only for multicasting, the only service supported by NetBIOS and not supported by SNA, for the compensations shown in the example. Other protocol combinations of MPTN, like SNA over TCP/IP, would require different compensations.

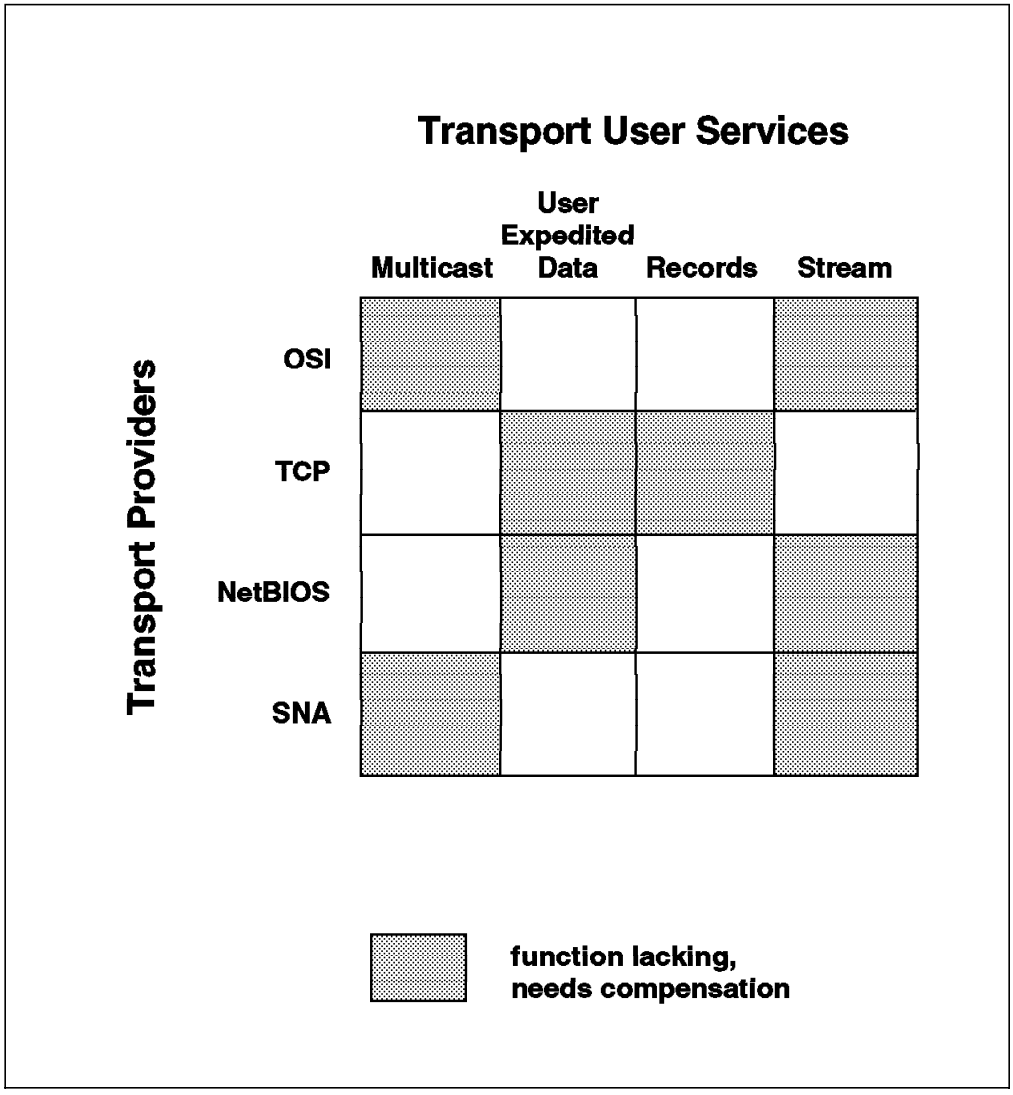


Figure 27. Compensations Provided by MPTN - Example

This is just a quick overview of the compensations required in MPTN. More detailed information is provided in 4.4.2, “The Protocol Compensator” on page 59.

On the other hand, there are protocols providing specific functions which cannot be terminated in the TLPB. For example, NETBIOS has some calls which return lower-layer information. The information returned by such calls must be simulated by the syntax mapper.

All services provided by the TLPB to the transport users will be shown in the form of *verbs* in 3.2, “Transport-Layer Protocol Boundary” on page 46. But now the services of the TLPB are shown in a more general way by grouping them into classes of functions, and the transport users are characterized according to these classes. Also, the basic mechanism for routing in the MPTN network is shown.

3.1.1 Connection-Oriented Services

The first class of TLPB functions is connection-oriented services. Connection-oriented services establish a logical connection between two partners (that is, two matching transport users) for the period they communicate with each other. Data transfer takes place over the connection in a sequenced and reliable manner.

- Connection Establishment

  The logical connection between two partners is established during a connection establishment phase. The transport users of some communications protocols send connection data (for example, a BIND in SNA) to specify the characteristics of the connection while other communication protocols (like TCP/IP) do not. MPTN supports connection setup for transport users that send connection data and allows the user to accept or reject the connection after examining the connection data. MPTN also supports connection setup for users that do not send connection data.

- Connection Termination

  During connection termination, a sequence of flows is exchanged that causes an existing connection to be released. MPTN supports connection termination for transport users that send termination data on these flows and those that do not send termination data. Some transport users expect to terminate one direction of the connection at a time, while others expect to terminate the entire connection at once. And there are transport users who expect the connection termination flows to stay in line behind data sent earlier, while others expect it to be able to overtake data sent previously. All of these termination semantics are supported by MPTN.

- Data Transfer

  Connection-oriented services must provide reliable, in-order delivery of data.

  – Reliable data transfer must be provided by the transport provider selected for a connection. If an unreliable data link or network layer protocol is being used, any data loss must be recovered by functions in a higher layer, such as the transport layer. These cases are handled in MPTN by using existing transport protocols such as TCP (over IP - which is connectionless and all error recovery must be done in the TCP stack which corresponds to the OSI transport layer) or OSI transport class 4 (which covers the case of transporting connection-oriented data flow over connectionless data link or network layer protocols).

  – In-order delivery of data requires that the destination transport user recognizes the sequence in which data were sent. MPTN inserts a sequence number in the MPTN header where required so that, at the receiving side, MPTN can pass the data to the transport user in the same sequence it was sent.

  Within connection-oriented data transfer there are other aspects that must be considered:

  – Stream and Record Data

    Some higher-layer protocols assume that data record boundaries are preserved by the underlying transport protocol. That means that a unit of data sent through the transport network is requested to be presented to the receiving partner as it was sent. Others send data as a byte

stream with no record boundary delineation. Both are supported by MPTN.

Since most transport protocols support one data type and not the other, compensation must be provided by MPTN for the unsupported data type.

– Maximum Record Sizes

Some transport protocols have a limitation on the maximum record size that can be transmitted. If a nonnative transport user has a maximum record size that is larger than that supported by the underlying transport protocol, compensation will be provided by MPTN; records will be segmented into smaller packets as supported by the transport provider, reassembled by the receiving MPTN function, and then the original record will be presented to the receiving transport user.

– Normal and Expedited Data

Some user applications and higher-layer protocols send expedited data. Such data may require flow control different from the flow control provided for normal data. Expedited data will not be delivered later than, but may be delivered ahead of, any normal data. MPTN can send expedited data and distinguish it from normal data when it is received. Expedited data is treated as either stream or record data according to the characteristics of the underlying transport provider.

When the expedited data function is not normally provided by a given transport provider (but is requested), compensation must be provided by MPTN. Compensation must also be provided when the maximum size of the transport user expedited data message is greater than that supported by the transport provider.

3.1.2 Connectionless Services

The second class of services of TLPB functions is connectionless services. Connectionless services is the mode of data transfer in which different units of data are passed independently through the network. Delivery is provided on a best effort basis, but undetected packet loss may occur, for example, if the network becomes congested. There is no guarantee that data units will be delivered in the same sequence that the sender issued them or that duplicates will not be delivered. The *datagram* is the basic unit of connectionless data transfer. This data packet has to carry all information required for routing from source to intended destination.

Different transport providers have different maximum datagram size limitations. The maximum datagram size expected by the transport user is supported. MPTN supports various connectionless modes:

- Unicast Datagram

  In this mode datagrams are exchanged between single partners.

- Multicast Datagram

  In this mode a datagram is sent to a group of transport users rather than to an unique user. The sender does not know who belongs to the group and need not be a member of the group. For example, a NetBIOS server on an SNA node could send a multicast datagram via SNA and an MPTN gateway to a group of NetBIOS users which are natively connected to a NetBIOS LAN.

- Broadcast Datagrams

A broadcast datagram is a datagram sent to all transport users having a certain generic address that have indicated a willingness to receive broadcast messages. MPTN treats the broadcast service as a special case of multicast.

3.1.3 MPTN Service Mode

The third class of TLPB functions are service mode services. Through MPTN gateways, single protocol transport networks (SPTNs) are connected and form the MPTN network. Data transmission in the MPTN network requires two levels of transport services working together to route the data from the source to the destination:

- The transport protocol of the transport provider in the SPTN
- The MPTN functions in the MPTN access nodes and gateways

Because a transport user may not be aware of which transport provider is selected for a requested connection or when sending a datagram to a partner, it cannot specify a service mode that is native to the transport provider. Instead, it may specify an MPTN service mode when requesting a connection or sending data.

The MPTN service mode indicates:

- High bandwidth is required
- Fast response time is required
- Secure service with high bandwidth is required
- Secure service with fast response time is required
- User-defined service mode
- No specific service mode is required

The selected service mode is passed in the MPTN header to the MPTN gateways involved and the MPTN access node when a connection is established or a datagram is sent.

To get the requested transport service within the MPTN network, the MPTN gateways define routing tables for each service mode so that requests can be forwarded to adjacent gateways in the network.

To get the requested transport service within each SPTN, MPTN functions map the MPTN service mode to a service mode of the transport protocol being used.

3.1.4 Selecting Transport Services

Before a transport user can communicate with other matching transport users, it must make itself known and describe its transport requirements to MPTN. The information about the transport user is recorded in a control block, a data structure which then serves as a transport endpoint. When creating the control block, MPTN must have information related to the type of communication that the control block will be used for; that is, which MPTN transport services the transport user requests.

The characteristics of the transport services offered by MPTN can be grouped into a general part which applies to all transport users, characteristics of connection-oriented services, and characteristics for connectionless services.

General Transport Characteristics

Service Type
: Indicates the type of MPTN services the transport user requires. The service type can be connection-oriented or connectionless.

Address Family
: The transport user specifies the address family that it will use. This refers to the format of the address that the transport user's application will register and expect. For example, if the transport user was running an FTP application, the format used would be that of an IP address.

Transport Provider Preference
: The transport user has the ability to restrict which transport providers may be used. The transport user may select one particular transport provider, indicate that any available transport provider will do, or prefer a particular transport provider based on service related criteria or on some policy based criteria.

Connection-Oriented Characteristics

Data Type
: Indicates the type of data to be sent on the connection, either stream or record.

Connection Data
: Indicates the maximum length of connection data in a connection request. A length of zero indicates that no connection data will be sent.

Connection Reply Data
: Indicates the maximum length of connection reply data sent by the transport user.

Close Type
: Indicates which types of termination the transport user can use. Four types are possible:

  - Simplex - Orderly
  - Simplex - Abortive
  - Duplex - Orderly
  - Duplex - Abortive

  Simplex close means just the send direction is terminated. Duplex close terminates both directions. Abortive means that data queued or in transit may be lost. Orderly means that data in the send direction will be delivered before the connection is terminated.

Connection Termination Data Length
: Indicates the maximum length of any connection termination data that may flow on the MPTN connection termination.

Maximum Record Length
: Indicates the maximum record length the transport user will send or receive.

Maximum Expedited Data Length
: Indicates the maximum expedited record length the transport user will send or receive.

Expedited Data
: Indicates whether the transport user will send expedited data.

Expedited Marking
: Indicates whether the transport user requires the ability to mark the position in the normal data where the expedited data is sent.

Partial Records
: Indicates whether the transport user is able to send or receive incomplete records.

Session Outage Notification
: Indicates that the transport user needs to be informed promptly when a connection is lost abnormally.

Connectionless Characteristics

Maximum Length of Data on Normal Datagrams
: Indicates the maximum length of normal data on datagrams that the transport user will send or receive.

Maximum Length of Data on Control Datagrams
: Indicates the maximum length of control data on datagrams that the transport user will send or receive.

Datagram Segmentation
: Indicates whether the transport user supports datagram segmentation.

3.1.5 Transport Protocol Characteristics for Specific Protocols

This section summarizes which transport characteristics apply for the SNA, NetBIOS, TCP, and OSI transport protocols. Compensation is required if a transport protocol does not support a service required by a transport user.

Table 3 (Page 1 of 2). Transport Services Supported by Protocols

	SNA	**NetBIOS**	**TCP/UDP**	**OSI**
Data Type	Record	Record	Stream	Record
Connection Data	Not supported (Note 1)	Not supported	Not supported	Not supported (Note 1)
Connection Reply Data	Not supported (Note 1)	Not supported	Not supported	Not supported (Note 1)
Close Type	Duplex and Simplex Abortive	Duplex Orderly	Simplex Orderly	Duplex Abortive
Connection Termination Data	Not supported (Note 1)	Not supported	Not supported	Not supported (Note 1)
Maximum Record Length	No Restriction	(128 K - 2) Bytes	Not applicable	Defined by underlying network layer
Maximum Expedited Data Length	86 Bytes	Not applicable	Not supported (Note 2)	16 Bytes
Expedited Data	Supported	Not supported	Not supported (Note 2)	Supported

Table 3 (Page 2 of 2). Transport Services Supported by Protocols				
	SNA	**NetBIOS**	**TCP/UDP**	**OSI**
Expedited Marking	Not supported	Not applicable	Supported	Not supported
Max. Length of Data on Normal Datagram	No Restriction	Implementation dependent	Implementation dependent	Defined by underlying network layer
Max. Length of Data on Control Datagram	No Restriction	Implementation dependent	Implementation dependent	Defined by underlying network layer
Note 1 In SNA data in connection requests and replies and in termination requests are formatted and do not support arbitrary user data. **Note 2** Although TCP/IP provides an urgent data mechanism (1 Byte), TCP/IP implementation does not guarantee the transport of expedited data in cases of severe congestion (TCP window size = 0); MPTN does not use this mechanism.				

3.2 Transport-Layer Protocol Boundary

The previous section described the transport-layer protocol boundary (TLPB) in terms of the transport services that it must support. This section provides a description of the TLPB as the transport user's logical interface into the MPTN network. The TLPB provides the user with the ability to access functions provided by components of the MPTN access node.

This interface is called a protocol boundary and not a programming interface in order to emphasize the importance of semantic as opposed to syntactic equivalence. Products implementing MPTN are not required to have syntactic equivalence to this protocol boundary, but must have semantic equivalence to it in order to ensure application portability.

MPTN defines two other protocol boundaries as a strictly convenient way to describe interactions internal to the MPTN access node and the MPTN gateway node. These are the below-specific protocol boundary (BSPB) and the gateway-specific protocol boundary (GSPB); the details of these protocol boundaries are beyond the scope of this publication and will be discussed only briefly.

3.2.1 Protocol Boundary Naming Conventions

The protocol boundaries described above use a call/return interface, with calls referred to as verbs. A given verb and its associated parameters describe a specific function of transport services that is invoked. Each time a verb is invoked, there is a *caller*, which is the component that initiates the verb, and a *callee*, the component that is the recipient of the call. For the TLPB, the components involved will be the transport user and the common MPTN manager (CMM). See Chapter 4, "MPTN Access Node" on page 53 for a description of the CMM.

Verbs for calls between the transport user and the CMM have names of the form M_XXXXX, with XXXXX representing some descriptive function name (verbs of the BSPB have names of the form P_XXXXX). Calls from the transport user to

the CMM are called downcalls and can be identified by a "_DC" appended to the name, whereas calls from the CMM to the transport user are called upcalls and can be identified by a "_UC" appended to the name.

Verbs can describe request calls and result calls. In request calls the caller asks for some action to be taken by the callee; in result calls the caller (which was the callee in the previous request call) informs the callee (which was the caller in the previous request call) whether or not the requested function completed successfully. Generally, but not in all cases, a downcall is a request call; that is, it is initiated by the transport user to request a transport service. However, asynchronous events from the network (like an incoming connection or termination request and the receipt of data) result in request calls "up" to the transport user.

All calls are *non-blocking*; that means the request call will always be returned immediately (of course, after some processing internal to the callee), even if the requested service is still in process. This is done to allow the caller to perform other work while the requested function is being processed. For example, the processing of a request to establish a connection involves not only local functions, but may require external data flows to locate the connection partner and the exchange of information with the destination transport user. This can take a considerable amount of time and the non-blocking model allows the transport user to do other work while the connection is being established.

When a function cannot be completed immediately, the request is returned to the caller with a status indicating that the request is pending. Once the function is completed (in our example, the connection is established or the connection establishment failed), a result call will be made to the caller of the request call. This result call has the same name as the call it completes, with the suffix "_DC" replaced by "_UC" or vice versa. In our example, an M_CONNECT_DC is returned immediately with a pending status and will be followed by an M_CONNECT_UC with a status indicating the success or failure of the connection request (see Figure 28).

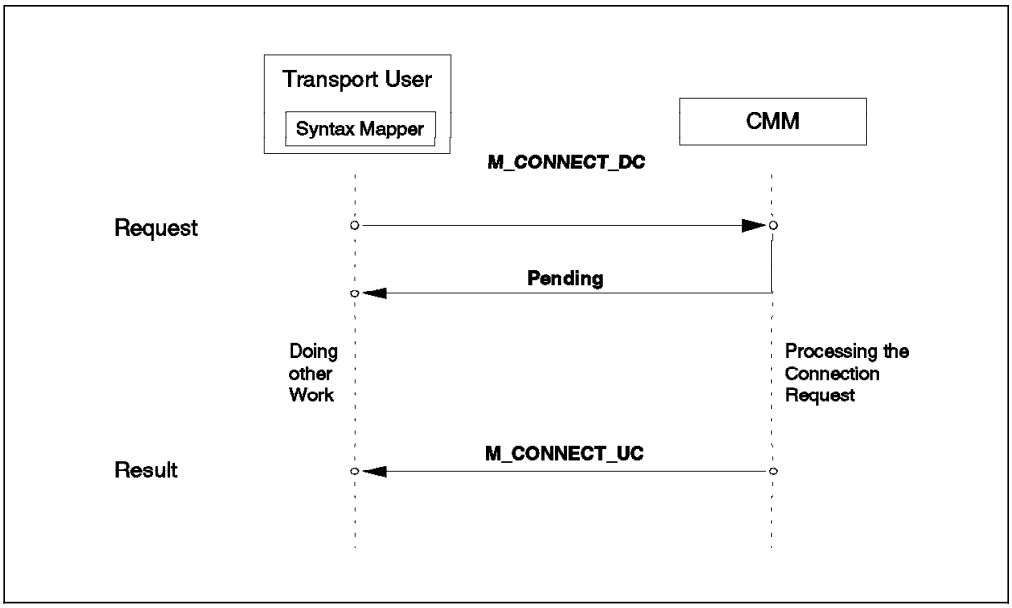


Figure 28. A Sample TLPB Call Sequence

3.2.2 TLPB Verbs

In the following sections we list all TLPB verbs grouped by functions and describe their functions briefly.

If you are familiar with AF_INET sockets calls, you will easily see close correspondences with the TLPB verbs.

3.2.2.1 Initialization

Before a transport user can communicate with other matching transport users, it must make itself known and describe its transport requirements. The transport user describes information about itself; MPTN then records it in a control block which serves as the transport endpoint.

M_CREATE_DC

With this TLPB verb the transport user identifies itself and describes its transport requirements, providing information related to the type of communication that the control block will be used for. This information includes the address family, the transport characteristics required, and any restrictions on transport providers that apply.

The address family is the format of the address that the transport user′s application will register and expect. In addition, the transport user has the ability to restrict which transport provider may be used with a particular control block. The transport user may select a specific transport provider or indicate that any available transport provider will do.

Details about the transport services requested include:

- The type of transport service that is requested (connection-oriented or connectionless)

  For example, a sockets transport user supporting a stream socket will choose connection-oriented service, but when supporting a datagram socket, will choose connectionless service.

- For connections:
  - The data type requested (stream or record)
  - The type of termination procedure required by the user
  - Whether the transport user needs to send expedited data
  - Whether the transport user requires the ability to mark the position in the normal data when expedited data are sent

3.2.2.2 Address Binding and Registration

Once a control block has been created by MPTN, it must be associated with an address by which the transport user is known to the network. If a transport user only initiates connections or only sends datagrams, it may indicate that its address should only be registered locally and not with an external directory server or an MPTN address mapper. See Chapter 6, “MPTN Address Mapping Services” on page 77 for further details of address registration and mapping.

M_BIND_DC

To associate its control block with an address, the transport user issues an M_BIND_DC indicating:

- The transport user's address to be registered
- Whether this address should be registered externally with a native directory server or an MPTN address mapper

If a transport user requires the ability to handle multicast datagrams, it must register its individual address and then join a multicast group by issuing an M_BIND_DC with the group address.

Note that in a network with MPTN access nodes supporting different address registration and resolution methods, a node has to support all mechanisms to be reachable from all other nodes. In this case, an address will to be registered using all supported methods.

M_BIND_UC

This verb notifies the transport user that the registration requested by a previous M_BIND_DC has completed and returns the status of the registration (successful or not).

M_UNBIND_DC

This verb requests the removal of the association between an MPTN control block and a transport-user address. If that control block is the last or only control block associated with that address, this verb initiates the deletion of any previous network directory or MPTN address mapper registrations.

3.2.2.3 Connection Setup

The setup for connection-oriented communications involves an active caller and a passive listener. A typical example is the case of client/server interaction where a client requests a connection with a server who is waiting to provide services.

M_CONNECT_DC

The active partner initiates an outgoing connection setup specifying:

- The address of the matching transport user that is the target of the connection
- The service mode of the connection
- Connection data that might be required by a transport user (for example, the BIND in SNA)

M_CONNECT_UC

This verb is used to inform the transport user of the result of a previous M_CONNECT_DC request. If the connection could be established the M_CONNECT_UC provides:

- User-specific data (for example, the BIND response of an SNA transport user)
- The modified connection characteristics
- The transport provider used for this connection

M_LISTEN_DC

In order to accept incoming connection requests from remote partners, the transport user must listen for connections by issuing this verb. The transport user will specify in this verb whether it wants to

listen for a single connection or multiple connections. When issued for a single connection attempt, the listen will be cancelled after the first incoming connection notification. If multiple connection attempts are chosen, then the listen will remain in effect until explicitly canceled by the user.

M_ACCEPT_UC

The M_ACCEPT_UC indicates to the transport user the arrival of a connection request and includes:

- The connection data sent by the initiator (if any)
- The address of both the initiating and the target transport user
- The service mode requested by the initiating transport user
- A pointer to a new MPTN control block which will represent the transport endpoint of the requested connection
- The connection characteristics requested for the connection by the initiating transport user (which can be modified by the destination transport user when returning the associated M_ACCEPT_DC)

M_ACCEPT_DC

The M_ACCEPT_DC is used by the transport users to indicate that they accept or reject a connection request presented in a previous M_ACCEPT_UC. When accepting the connection request, the transport user includes the connection reply data (if any) and any modified connection characteristics.

3.2.2.4 Connection Termination

Connection termination can be initiated by either partner of an established connection. The partner will then be informed that the connection is going to be terminated.

M_CLOSE_DC

This verb terminates an already established connection in the manner specified by the close type indicated which may be:

- Simplex - Orderly
- Simplex - Abortive
- Duplex - Orderly
- Duplex - Abortive

Simplex close terminates the send direction only, whereas duplex close terminates both directions; abortive close means that data queued or in flight may be lost, whereas orderly means that data in the send direction will be delivered before the connection is terminated. Termination data (if present) will be sent to the partner with the termination request.

M_CLOSE_UC

This verb returns the result for a previous M_CLOSE_DC.

M_CLOSED_UC

This verb notifies the transport user of a connection termination initiated externally to the local node and includes the type of termination requested and transport user-specific termination data sent by the partner initiating the termination.

3.2.2.5 Connection-Oriented Data Transfer

Connection-oriented data transfer between a pair of transport users provides for error-free, in-sequence delivery of data sent by either of the two connected partners.

M_SEND_DC
: This verb is issued by a transport user to send data on a connection. The sending transport user may indicate that the data being sent is expedited data or, if the data is an incomplete record, more data will follow.

M_RECEIVE_UC
: This verb indicates to the transport user that data has arrived on a connection. It indicates whether data is normal or expedited and whether the data being passed to the transport user completes a record (if the transport user supports partial records).

3.2.2.6 Connectionless Data Transfer

A datagram is the basic transmission unit for connectionless communication. MPTN supports the sending and receiving of datagrams over various transport providers. The TLPB provides a non-blocking datagram interface for use by transport users. A datagram can be sent to an individual address or a group address.

M_RCVDG_INIT_DC
: A transport user wishing to receive datagrams issues this verb with an indication of whether he is willing to receive:

  - Broadcast datagrams

    Datagrams will always be accepted for the transport user′s individual address or its group address (if applicable).

  - Multiple datagrams

    If multiple datagrams are not desired, listening will be stopped for datagrams after one has been received.

M_SEND_DG_DC
: Sends a datagram to either a specific (unique) destination, or to all the members of a group. The destination address type indicates, whether a datagram is unicast or multicast. It is also possible to send a broadcast datagram to all transport users that have indicated that they are willing to receive broadcast datagrams. In the M_SEND_DG_DC verb the transport user indicates:

  - The service mode requested for the datagram
  - Whether the transport user is attempting to resend a previously sent datagram that could not be delivered to its target
  - Whether the datagram is complete or a partial segment

M_SEND_DG_UC
: If a request to send a datagram requires address resolution using an external directory server or MPTN address mapper, this verb is used to return the status of the previous send request.

M_RCV_DG_UC

This verb informs the transport user that a datagram from a remote transport user has arrived and indicates: .

- The address of the sending transport user
- Whether this is a datagram segment
- Whether the datagram was received from a broadcast

Chapter 4. MPTN Access Node

Figure 29 shows a high-level picture of an MPTN node. The problem of translating the transport semantics used by a transport user into the semantics supported by a nonnative protocol stack of a transport provider can be addressed by modifying each transport user to include a *syntax mapper* which transforms the transport requests native to the transport user into TLPB primitives.

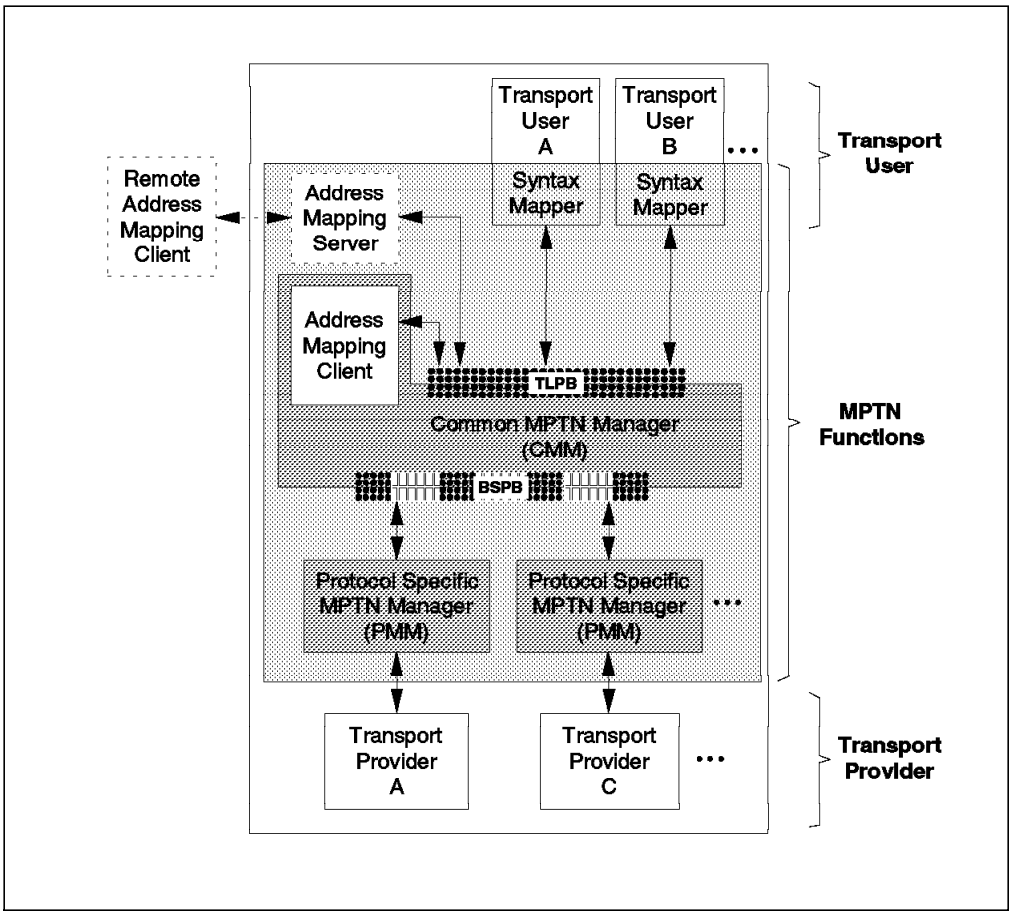


Figure 29. MPTN Access Node - High-Level Overview

The figure shows MPTN transport users that request services across the TLPB, which are in turn serviced by MPTN transport providers that provide these services. The common MPTN manager (CMM) component provides services such as the selection of compensations, general address mapping, connection establishment, connection-oriented data transfer, and datagram routing. Service requests from the transport users are forwarded to the proper protocol-specific MPTN manager (PMM) which maps those requests to the protocol supported by the specific transport provider across its native API. Protocol compensation is required when a transport user requires transport services that are not natively supported by the selected transport provider.

An additional function provided by the CMM is address mapping services. The address mapping services function may be performed purely locally within the node as would be the case if algorithmic mapping of addresses between those of the transport user and those of the transport provider were the address mapping

technique or the function may be performed with the aid of an address mapping server, a function which is shown as present within the node but which may reside in another node. When an address mapping server is used there is a complementary entity, the address mapping client, operating from the address mapping services function within CMM. Since the address mapping server function is designed to be a shared facility, the figure shows other address mapping clients which can be using the services of the address mapping server in the node illustrated. The address mapping client and server functions may, in fact, be present in either MPTN access or gateway nodes.

The TLPB provides a common set of transport services to satisfy the requirements of existing transport users. The discrepancies between the functions requested by the transport user (and offered by the TLPB) and the actual services supported by the selected transport provider is addressed by a universal set of *MPTN protocol compensations*.

The MPTN architecture also provides *address mapping* functions so that an application, through the transport user, can identify itself and its partner(s) using addresses belonging to its native address type, even though the underlying transport provider uses addresses of a different type.

4.1 Access Node Structure

The components of an MPTN access node are divided into those that *use* the transport services and those that *provide* transport services. Some of the components involved in providing transport services also use the TLPB to talk to their counterparts in other nodes.

The *common MPTN manager (CMM)* represents the portion of the MPTN functions that can be performed independently of any particular transport provider protocol. Examples of CMM functions include registering transport-user addresses to make them accessible over nonnative protocols, selecting a transport provider to create a connection between two transport users, and reporting events to network management components.

CMM includes:

- A function that processes all TLPB verbs, because the state rules for these verbs are independent of any transport protocol.
- *Address mapping services* that performs common address resolution functions. If address mapping services requires the use of an address mapper, this component, behaving as an address mapping client, communicates with one or more address mapping servers to register transport-user addresses together with the one or more corresponding transport-provider addresses or resolve transport-user addresses to transport-provider addresses. To communicate with an address mapping server, the address mapping client functions as a transport user; that is, it issues TLPB verbs to communicate with the address mapping server.
- Services that participate in datagram routing, connection establishment and takedown, network management, and other support services required to run multiple transports.

An *address mapping server* is an optional component that provides address resolution services to local and remote *address mapping clients* in CMMs using

MPTN services. For information about address mapping services, see 6.3.1, “MPTN Address Mapping Service” on page 81.

The *protocol-specific MPTN manager (PMM)* performs any MPTN management functions that are specific to a particular transport provider protocol type. There will be a distinct PMM for each transport provider installed in an MPTN node. Thus, there will be a PMM supporting the SNA transport provider and a PMM supporting the TCP/IP transport provider, if SNA and TCP/IP were used as transport providers in an MPTN access node.

4.2 Transport Users

TLPB is intended for use by programs that provide communications facilities for end users, such as:

- Communications resource managers. The Networking Blueprint identifies three specific communication services: conversation, remote procedure call, and messaging and queuing.
- Other special high-performance support programs (for example, LAN Manager).
- Higher layer protocols that support various end user functions and APIs, such as SNA LUs and OSI.
- Existing end user APIs that are mapped to TLPB, such as NetBIOS end user interface (NetBEUI), AF_INET sockets, or XTI.

A transport user identifies itself and describes its transport requirements to the CMM using the M_CREATE_DC verb. The information related to the requested type of communication provided with this verb includes the address family, the transport characteristics required, and any restrictions on transport providers that apply. The address family that the transport user will use refers to the format of the address that the transport user′s application will register (using the M_BIND_DC verb) and expect. For example, if the transport user was running an FTP application the format used would be that of an internet address.

Details about the transport services requested include whether the type of transport services requested is connection-oriented or connectionless, and, in the case of the connection-oriented transport service, the data type (stream or record) and the termination type indicated, and whether expedited data must be supported.

4.2.1 Transport User Types

Figure 30 on page 56 shows the three different types of transport users. The three different types are shown with different sized boxes to indicate differences in the amount of function involved:

- Those communication users that use the TLPB directly. This box is the smallest since it represents no additional function between the communication user and the TLPB. No syntax mapper is needed.
- Those that support user APIs with functions very similar to the TLPB functions, that is, primarily transport functions. These boxes are middle-sized since they represent transport users that require only syntax mapping between the API and the TLPB.

- Those that support user APIs that require additional protocol above and beyond that provided by the TLPB. These are the largest boxes since they require protocol in addition to syntax mapping. Examples include LU 6.2 protocols, OSI TP protocols, and LU 2 protocols.

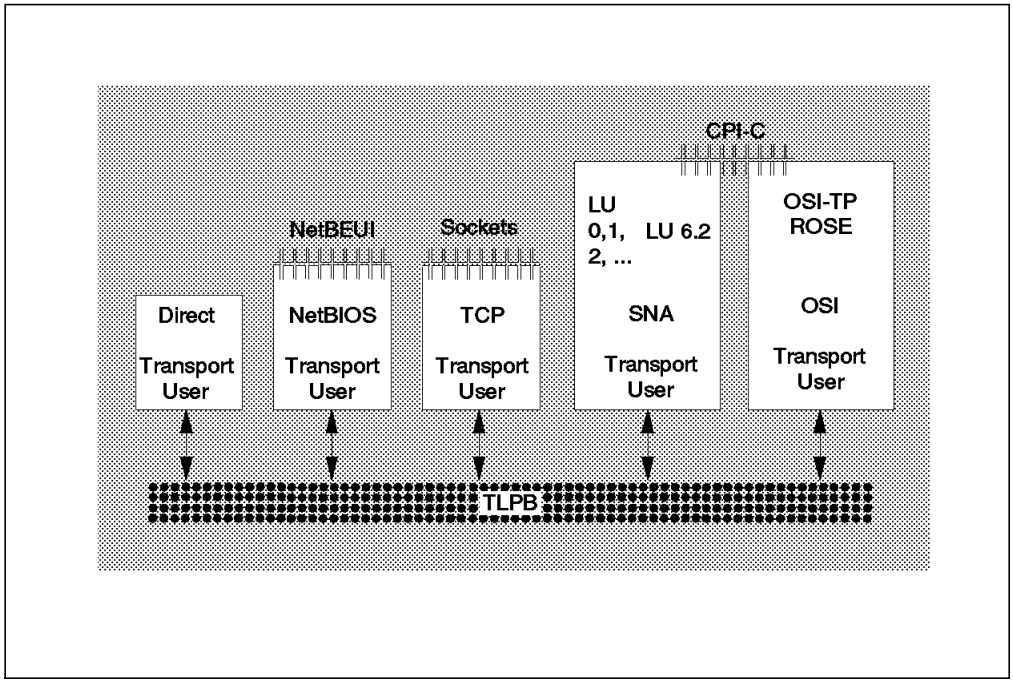


Figure 30. Transport User Types

Let us discuss each transport user type separately:

- *Direct User* represents the transport user that does not implement a *protocol-specific* function above the TLPB. It does not have an API since their communication users run directly on the TLPB. Examples are MPTN internal functions such as the MPTN address mapping client and server components.
- *TCP* and *NetBIOS* implement syntax mappers that allow TCP/IP programs or NetBEUI programs to run over the TLPB.
- *OSI* and *SNA* both represent additional protocol. For example, half-duplex LU 6.2 includes conversation state management in addition to confirmation and error processing, over and above the transport protocols. OSI includes the functions from the session, presentation, and applications layers.

  CPI-C, an API provided to communication users, is supported by both OSI-TP and LU 6.2. LU 6.2 also supports the APPC API.

  In addition, the SNA transport user component includes protocol for running all LU types.

4.3 Common MPTN Manager

MPTN manager components are concerned with the communication flows necessary to connect the transport users above TLPB with the transport functions below TLPB. The *common MPTN manager (CMM)* implements management functions that can be performed the same way for all transport protocols, in contrast to the *protocol-specific MPTN manager (PMM)*, which performs any MPTN management functions that are *specific* to the protocol types of a particular transport provider.

4.3.1 Transport Endpoints

Before a transport user can communicate with other matching transport users, it must make itself known and describe its transport requirements to the CMM. The information about the transport user is recorded in a control block, a data structure which defines a *transport endpoint*. In order to be accessible to the network, a transport endpoint must have an address associated with it.

The transport user causes the control block to be created by issuing an M_CREATE_DC verb. The CMM uses this information when it provides services to the user.

The CMM maintains the following information about each transport endpoint assigned to a transport user:

- A transport provider preference list

  The list is maintained in preference order to be used to satisfy connection or datagram requests. The list is generated by pruning the list of transport providers in the node based on the M_CREATE_DC parameter indicating which transport providers may be used, and then reordering the remaining list based on which can best provide the transport services associated with the user.

- Service types that will be requested, such as connection, connectionless, expedited data transfer, stream or record

  Using this information, the CMM can calculate connection characteristics and select the best transport provider for the services requested by the user. Thus, the CMM would not select SNA as the transport provider for a transport user that requests connectionless service if the destination can be reached using a transport provider that natively supports datagram services.

- Function to be invoked when the transport user's connection terminates

- Transport-user and transport-provider addresses associated with that transport user

4.3.2 Communication with an MPTN Address Mapping Server

As described in Chapter 6, "MPTN Address Mapping Services" on page 77, the address mapping services component of the CMM can use several different techniques to register and resolve addresses, including interaction with an MPTN address mapping server. In this case the address mapping services component behaves as an address mapping client.

Some transport providers can only be served by an address mapping server, while others may be served by another address resolution technique, such as

enhanced use of a transport provider specific directory or algorithmic address mapping.

The address mapping services component of the CMM, the address mapping *client*, runs as a direct user of the TLPB, and when it determines that it needs services of an address mapping server, it uses TLPB verbs to communicate with the address mapping server.

4.4 The Protocol-Specific MPTN Manager

The internal structure of the *protocol-specific MPTN manager* is shown in Figure 31. The PMM is a collection of components that make up the MPTN support related to a *specific* transport protocol.

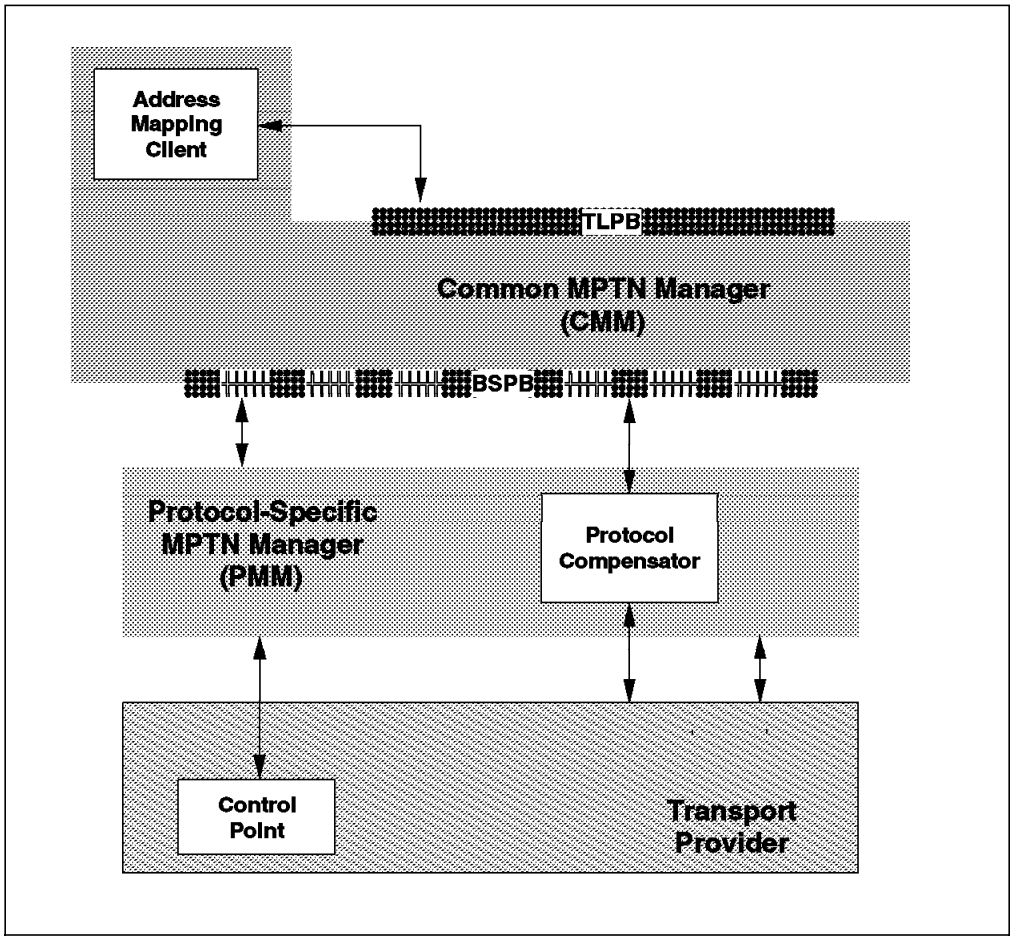


Figure 31. PMM Components

The PMM creates an instance of a *protocol compensator* for every connection that allows a transport user to run over the transport provider, compensating for the differences between the services requested by the transport user and the services provided by the transport provider.

4.4.1 PMM Functions

The PMM performs protocol-specific management functions. The PMM is involved in advertising names of transport users and opening connections. As shown in Figure 31 on page 58, the PMM and the CMM interact. For this interaction the *below-specific protocol boundary (BSPB)*, which is the protocol boundary between the CMM and the PMM ("below" the CMM), is used. The PMM and the protocol compensator interact with the transport provider using whatever internal interface it provides. The PMM will map the BSPB verbs to primitives that are expected by the transport provider's API.

When a transport user's address is advertised by an enhanced use of a native directory function, the PMM forwards the request from the CMM to nodes supporting the native directory function.

In setting up a connection, the PMM accepts a request from the CMM to set up a connection and generates a native connection setup request to the transport provider. At the passive end of the connection, the PMM receives a request from the CMM to listen for connection requests and translates them to the transport provider, if applicable. For example the TCP PMM would issue a passive Open call to TCP, whereas no special call would be necessary to listen for an SNA connection request. The CMM and the PMM also send connection data, if requested by the transport user.

4.4.2 The Protocol Compensator

The protocol compensator is the component that handles service requests from the transport user when it has to use a transport provider that does not provide all the services it requires.

When the CMM sets up or accepts a connection, it determines from the connection characteristics of the two transport users, which compensations are required. The PMM then, on request from the CMM, creates an instance of a protocol compensator for that connection. Although the generalized set of compensations are selected by the CMM, the protocol compensator, which is tailored for the compensations needed for a connection, is a function of the PMM.

The major task of a protocol compensator is to compensate for functions requested by the transport user that are unavailable in the transport provider's protocol stack. The protocol compensator accepts verbs from nonnative transport users, provides compensation for various functions by adding appropriate MPTN headers, and issues primitives to the transport provider to perform the communication function requested.

4.4.3 Protocol Compensation

Protocol compensation is required when the transport user reaches his communication partner via a transport protocol that does not match the native protocol. As we have already seen in 3.1.5, "Transport Protocol Characteristics for Specific Protocols" on page 45, no transport protocol supports every transport characteristic. For unsupported characteristics of the native protocol stack of the transport user, MPTN must perform compensation. Most of these compensations are performed by the protocol compensator component. Exceptions include the compensation required for datagrams and those required for transferring connection data. The CMM knows the transport characteristics of the transport user and it knows the characteristics of the transport provider.

In order to provide compensation only for those functions that the transport user requires and the transport provider does not support, the “compensation package” for a specific (user,provider) pair is selected dynamically from a universal set of compensations. It is *not* predefined for specific pairs of mismatched protocols. Therefore, the addition of either a new transport protocol type or a new user type requires only the normal analysis and selection of existing compensations, not necessarily the addition of an entirely new compensation.

4.4.3.1 Compensation Mechanisms

Protocol compensation functions are performed by at least two protocol compensators involved in an MPTN communication. For multicast and broadcast messages, more than two protocol compensators will normally be involved. The following discussion considers that two transport users will communicate using a given transport protocol. The CMMs in the two nodes exchange information about the characteristics of their respective transport users. If certain transport characteristics do not match, an instance of a protocol compensator is created at each end and informed about the mismatched characteristics. Thus the protocol compensators at both ends agree about the types of compensation-related information they should send to each other.

Most of the compensation functions are performed by adding special headers to the data stream, but some compensations do not require headers or others may require more than just a header. When a compensation that adds special headers to the data stream is required, a header is attached to *every* packet on the data stream, even if the data in that packet does not require compensation. This is necessary so that the protocol compensator in the partner node can distinguish headers from data.

The method of compensation is described for the following transmission characteristics (for information on transport services supported by specific protocols, refer to Table 3 on page 45):

- **Connection data**: Connections between user applications are set up as virtual connections over transport provider connections. Once the underlying connection is set up (between the two CMMs), special packets containing the transport-user addresses, the connection characteristics, and the connection request and response data (if any) are exchanged between the two CMMs. Any connection data sent by the transport user initiating the connection is extracted from the special packet by CMM at the passive end of the connection request, and passed to the transport user to which it was sent. Any reply to the connection data is handled in the same way.
- **Termination data**: A special header is inserted by the sending protocol compensator to identify the termination data. The receiving protocol compensator removes the header, extracts the termination data, and passes it up to the transport user.
- **Full-duplex over half-duplex**: A technique used for this is to set up two half-duplex connections, one that is always in “send” state and the other always in “receive” state at a given end of the connection. The compensation required includes coordinating take down of the two connections when either a normal or abnormal termination occurs.
- **Record boundaries**: A special header identifies the beginning of the record and the length, enabling it to be reassembled by the receiving protocol

compensator even if it has been split by a stream-oriented transport protocol.

- **Expedited data**: A special header identifies the expedited data and its length. It is sent in the normal data stream and extracted by the receiving protocol compensator. The advantage of sending expedited data in the normal data stream is that it does not arrive after the normal data. In cases of congestion, however, it is also sent as a connectionless message in order to assure delivery.
- **Positional correlation between expedited and normal data**: When the transport provider does not support expedited data, positional correlation is automatically provided by the previous solution since expedited data is sent on the normal data stream. When expedited data is supported by the transport protocol, positional correlation can be provided by adding a special header containing the current pointer to the normal data stream to the expedited data. If the header and the expedited data are longer than the maximum amount of expedited data allowed by the transport protocol, segmentation has to be performed; see next paragraph.

  Positional correlation is required to implement the *out-of-band* function of TCP. One use of positional correlation is in the TELNET application where all normal data up to the position at which a control function (the interrupt control function) was entered is discarded but following data is not. Clearly the function of the control sequence is to ensure that any data received, but not yet processed, is discarded.
- **Expedited data length mismatch**: There are two cases to be considered:
  - Subsequent expedited sends cause the preceding expedited data to be lost if it is not received fast enough. In this case, a header with the length field is added before the expedited data. As much of the composite packet is sent as expedited data as allowed by the underlying transport (for example, 1 byte in TCP/IP), and the rest is sent in the normal data stream. The receiving protocol compensator receives urgent notification about the header and flushes the normal data stream to receive the rest of the expedited data.
  - Subsequent expedited sends do not cause the preceding expedited data to be lost. In this case, a header with a length field is added before the expedited data. The composite packet is segmented into packets whose sizes are not greater than the maximum amount of expedited data allowed by the underlying transport, and each segment is sent as expedited data. Note that segments sent after the first one do not contain headers; a solution that works if the underlying transport protocol guarantees ordering of expedited data.
- **Datagrams over connections**: Implementations of MPTN function in the transport provider network may select whatever way is most suitable. These include the OSI connectionless over connection-oriented standard and sending one way brackets over SNA sessions.

  When datagrams are sent over sessions, a common implementation choice will be to retain the session for a period after the datagram has been sent in the expectation that more datagrams may follow in the near future. The period may well depend upon the relative costs of retaining the session over time in comparison with the costs of reestablishing the session.
- **Datagram length mismatch**: Connectionless data is directed to well-known (reserved) transport-provider addresses. Every unicast or multicast

datagram exchanged between transport users on nonnative providers already contain headers with additional information about source and target names so that the datagram can be delivered to the correct transport user. When there is a length mismatch, the datagram is segmented and an additional header is added to each segment. The segment is sent out as an individual datagram. Additional information in the segment header assures proper reassembly.

- **Record length mismatch**: This is handled by segmentation, with each segment header containing a field that identifies if the record is complete. Since ordering is guaranteed, the segments will arrive in order and the transport user will be informed when the record is complete.
- **Orderly close over abortive close**: With abortive close, the transport protocol does not make any effort to flush data in transit when it receives a request to close a connection. When a transport user requests orderly close, the sending protocol compensator inserts a packet with a special header into the normal data stream. When the receiving protocol compensator receives this packet, it issues an acknowledgement because it can be sure it has received all the normal data that preceded the close. The partner can return the orderly close to the transport user since all normal data has been sent. Then the underlying transport provider connection can be closed.
- **Simplex close over duplex close**: When the first transport user issues a simplex close, a special packet is sent by the protocol compensator to its partner. The partner notes the receipt of this packet but does not close the connection. Thus, the second transport user can still send data over this connection. When the second transport user also issues a simplex close, its protocol compensator issues a (duplex) close to terminate the connection.
- **Duplex close over simplex close**: In this case, since the two protocol compensators expect their transport users to request duplex close, this compensation is performed without inserting any additional information in the data stream. When one user issues a duplex close, its protocol compensator issues a simplex close to terminate one end of the transport connection. The receiving protocol compensator also issues a simplex close to complete the connection termination.
- **Session outage notification:** If the transport provider used for a connection does not notify the connection end points when a link or node fails on the physical path used for the connection, the PMMs regularly exchange "keepalive" messages to check that the partner still can be reached. In case of a failure on the connection path, this keepalive message will time out waiting for a response. The PMMs will close *all* MPTN connections to the addressed node and the associated CMMs are notified. This notification is passed to the transport users with appropriate protocol flows.

4.5 Connection Establishment and Datagram Routing

In this section we describe how an M_CONNECT_DC request issued by a transport user is linked to a particular transport provider protocol on the active side of the connection, and how the arriving connection request is linked to the matching transport user on the passive side. We also describe the CMM actions in routing datagrams, since these are strongly related to the actions involved in establishing connections.

4.5.1 Connection Establishment - Active Side Open Actions

When the transport user opens a connection, it identifies its partner by a transport-user address. The CMM that processes the M_CONNECT_DC verb performs the following functions:

- Select a transport provider over which to route the connection.

  On the originating side of the connection, CMM is responsible for selecting the transport provider to provide the transport service. It does so by arranging the potential transport providers into preference order and trying each one until it succeeds or exhausts the list. Matching characteristics requested for the connection or input from the transport user can be used to adjust this list.

  A native transport provider, if available, is selected by the transport user before requesting a connection over the TLPB. Such native connections will not use any MPTN functions at all.

- Discover the transport-provider address of the destination.

  If a nonnative transport provider has been selected, the CMM invokes address mapping services to find a transport-provider address for the destination. First, it searches its cache for a match. Address mapping services returns a node address for the partner. The CMM *derives* the node address.local address tuple by appending the well-known local address provided by the transport provider at installation time (null for NetBIOS and SNA).

- Determine the connection characteristics based on transport user characteristics and transport provider characteristics.

  This includes whether a protocol compensator is required and whether MPTN headers are required.

- Invoke the transport provider to establish a connection.

  This is the underlying transport provider connection used to carry the MPTN connection.

- Exchange MPTN_Connect and response to MPTN_Connect with the partner CMM.

  Two CMMs transform a transport provider connection into an MPTN connection by exchanging MPTN_Connect and response to MPTN_Connect messages over it. The CMM on the active side builds the MPTN_Connect, which includes:

  – The transport-user addresses of the source and destination

  – The connection data provided, for example, by an OSI or SNA transport user

  – The connection characteristics that the CMM passed to the transport provider

  – Whether MPTN headers are used on this connection

  The passive CMM parses the MPTN_Connect to extract the information and builds the response to MPTN_Connect format, into which the (destination) transport user's connection reply data can be inserted.

4.5.2 Connection Establishment - Passive Side

On the passive side, a transport user that wants to accept connections issues an M_LISTEN_DC verb. When the PMM receives a connection request over its native transport provider it will participate in the native transport provider's connection setup protocol and wait for the arrival of the MPTN_Connect command sent by the initiating PMM once the native connection is established. PMM then forwards the MPTN_Connect command to the node's CMM indicating the origin and destination transport-provider addresses, and an indication of which transport provider is being used for the connection. The primary job of the CMM then is to match the incoming connection request to M_LISTEN_DC verbs posted by transport users. The target transport user is then informed about the connection request by an M_ACCEPT_UC including the following information:

- The connection data sent by the initiator, if any
- The transport-user addresses of the initiator and the target transport user
- The service mode requested by the initiating transport user
- A pointer to the transport endpoint control block to be used for this connection
- The connection characteristics requested for the connection by the initiating transport user

The transport user will then return an M_ACCPEPT_DC either modifying or accepting the connection characteristics and providing the connection reply data if required in the transport user's protocol. On request from CMM, PMM will then create and send an MPTN_Connect_Rsp back to the originating node. The transport user's connection is now ready to transport data.

4.5.3 Connection-Oriented Data Transmission Considerations

Data transfer between a pair of connected transport users involves one partner issuing an M_SEND_DC verb. The M_RECEIVE_UC then informs the transport user on the receiving side that data has arrived.

Connection-oriented data transfer must be reliable. This requires that data be delivered to its destination error-free and without any loss. Any error recovery must be performed below the TLPB, so that the transport user is not aware of any recovery procedures.

Data transfer is in-sequence. Data sent using multiple M_SEND_DC verbs must be delivered to the destination in the same sequence it was passed to the TLPB.

The data type can be stream or record for a single connection. The data type used by a transport user is defined in the transport characteristics when initializing its interface to the TLPB with an M_CREATE_DC. This implies that a transport user can use either record or stream data format.

When *stream* data type is used, no attempt is made to preserve any message boundaries.

When *records* are transmitted, the transport user must indicate the end of the record and whether the record is segmented. A segmented record is transferred between TLPB and transport user in multiple calls. On the sending side, the transport user sends the segmented record within more than one M_SEND_DC. On the last M_SEND_DC of the segmented record, the transport user indicates

that the record is complete. When a transport user receives a record, it can receive this record in more the one segment, which means in more than one M_RECEIVE_UC. When a transport user is able to receive or send segmented records it indicates this in the M_CREATE_DC. The following figures show examples for connection-oriented data transfer either with segmented records or with data streams over different transmission sizes.

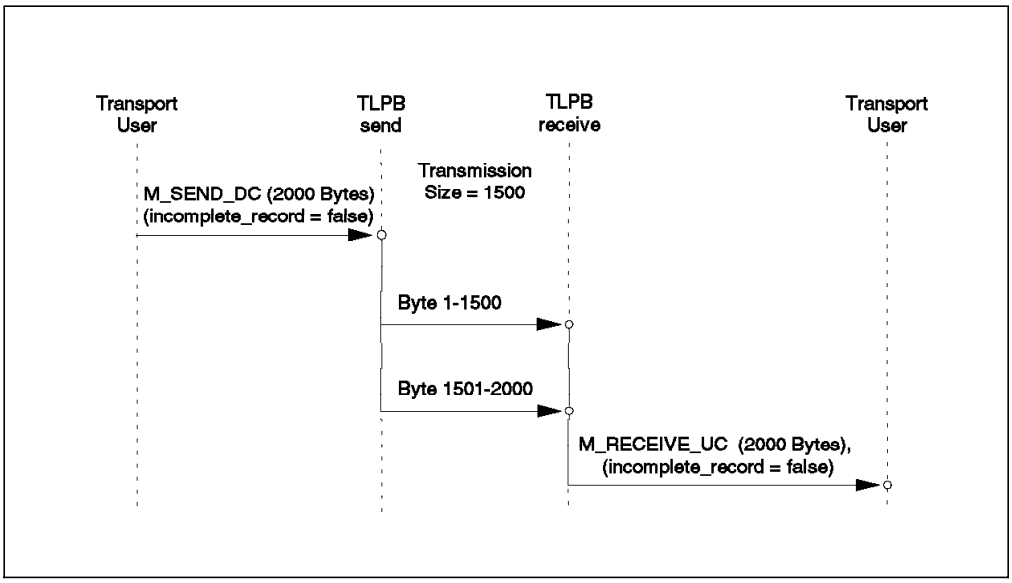


Figure 32. Record Data Flow in an MPTN Connection for Segmented Records. Segment reassembly is not supported by the transport user.

If a transport user cannot support segmented records, MPTN must do the reassembly in cases where a segmented record arrives for this user. Even if the sending transport user has sent the record in one M_SEND_DC, it is possible that the record is segmented (for example, the transmission size on an intermediate link is smaller than a specific record). This is illustrated in Figure 32.

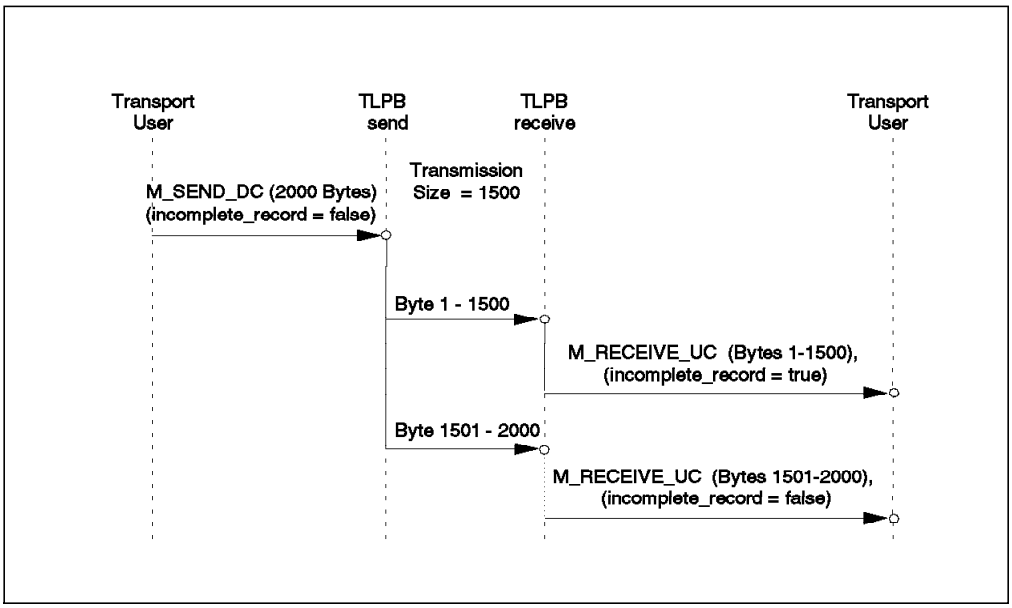


Figure 33. Record Data Flow for Segmented Records with Segment Reassembly. Segment reassembly is supported by the transport user.

If a transport user can support segmented records, MPTN passes the segments of a record to the transport user in multiple upcalls. The last upcall for the segmented record indicates that the record is complete. This is shown in Figure 33.

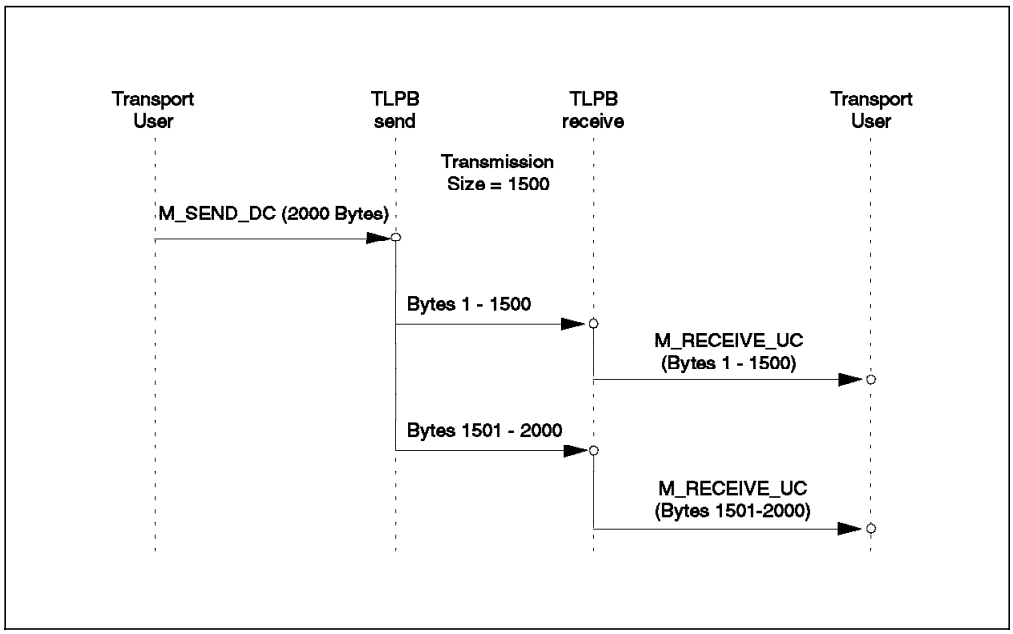


Figure 34. Stream Data Flow in an MPTN Connection. (Transmission buffer size = 1500)

When data streaming is used, no attempt is made to preserve any message boundaries. The different data transfers for 2000 bytes of streaming data are shown in Figure 34 and in Figure 35.

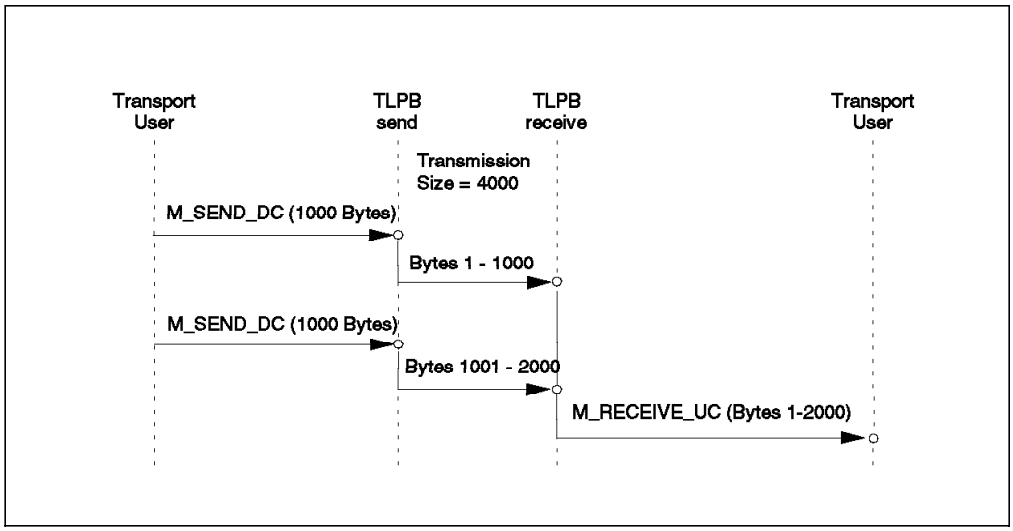


Figure 35. Stream Data Flow in an MPTN Connection. (Transmission buffer size = 4000)

In Figure 34 the 2000 bytes are sent at once, but the transmission size requires a data split, so the data arrive in two parts. The upcall for the first part is already initiated before the second part has arrived. A second upcall is required to transfer all the data to the transport user. In Figure 35 the 2000 bytes are sent with two M_SEND_DC calls. The transmission sizes of 4000 bytes would allow all the data to be transferred in one send, but the second send would take too long. So the first data part is sent and then the second. But CMM at the

receiving side tries to fill up the receive buffer and wait long enough to receive the second part also before the upcall to the transport user is performed.

The previous two figures for data transfer in data streaming mode show just two cases of many possibilities of how a data stream could be transmitted. How data is collected and when the transport user receives the upcall for the data which has arrived depends on the transmission provider used for the actual data transfer. CMM could wait for a short period if there is more data arriving and try to fill a buffer before the upcall to the transport user is initiated.

4.5.4 Routing Datagrams

This section provides a brief summary of the CMM actions for datagram routing, both unicast and multicast.

Source Side: Every transport endpoint with a service type of connectionless is associated with a *preference list* of nonnative transports providers that can be used to send datagrams for that transport user. The list is calculated at M_CREATE_DC time. The transport user may restrict the set of transport providers that can be used for datagram routing. The datagram characteristics, including the multicast indicator and the requested maximum datagram length, affect the order of the list. For example, if a user requests multicast, the list is ordered so that transport providers that support multicast natively are more likely to be used than those that do not.

When a transport user issues an M_SEND_DG_DC verb, the CMM creates a preference list of transport providers by making modifications to the preference list associated with transport user′s transport endpoint control block.

The CMM that processes an M_SEND_DG_DC verb performs the following functions:

- Select the transport provider over which to send the datagram.
- Discover the transport-provider address of the destination.
- Format an MPTN datagram message that includes the transport-user addresses of the source and destination.
- Invoke the transport provider to send the datagram.

Destination Side: A transport user wishing to receive datagrams issues an M_RCVDG_INIT_DC to the CMM together with an indication whether it wishes to receive broadcast datagrams or multiple datagrams. When a datagram arrives, the CMM passes the datagram to the target transport user identified by the target address in the datagram in an M_RCV_DG_UC together with the origin and target transport-user addresses and an indication whether the datagram received was from a broadcast.

If the datagram has been segmented, the CMM, if requested to do so in the previous M_RCVDG_INIT_DC, will reassemble the datagram and forward the complete datagram in one M_RCV_DG_UC, or else the transport user has to reassemble the datagram and each segment is forwarded in an M_RCV_DG_UC separately.

Chapter 5. MPTN Gateway Node

This chapter describes the function, structure, and components of the MPTN gateway node. We show what functions are provided by the MPTN gateway and how it connects networks that have differing transport protocols into one large network.

5.1 MPTN Gateway Function

The primary function of an *MPTN gateway* is to connect individual single protocol transport networks (SPTNs) to form an integrated heterogeneous network. An SPTN is a collection of physically connected nodes that implement a common transport protocol.

MPTN gateways provide interconnectivity at the transport layer. This implies that the SPTN transport and lower-layer protocols are terminated at the gateway. User data is relayed over transport-layer connections between adjacent gateways, or between gateways and end systems. Figure 36 shows a logical view of an MPTN network.

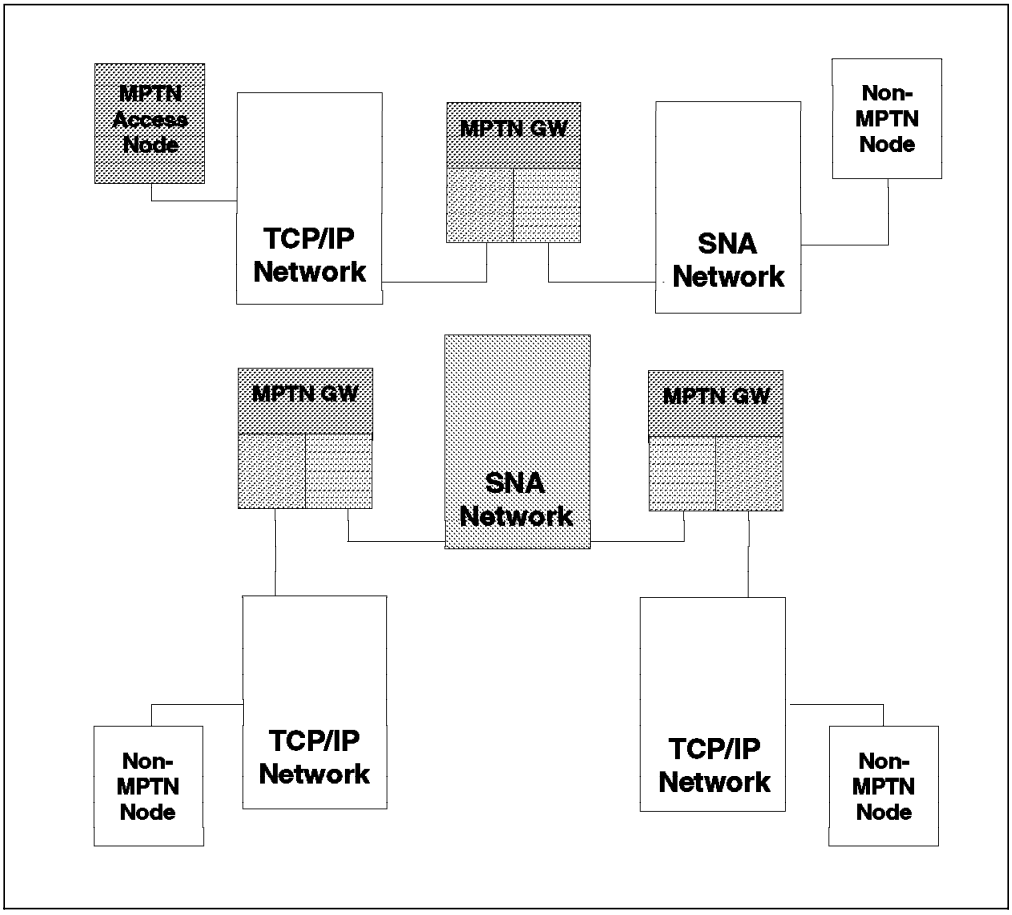


Figure 36. Logical View of an MPTN Network

For protocols shown in Figure 36 the MPTN design is general; the MPTN gateway is designed to make implementation of new protocols easy.

Each SPTN in the above figure may itself be a composite network (described previously in 2.4, “Network Types and Connections” on page 22), requiring the gateway also to function as a *native routing gateway* so that the gateway can participate in the routing protocol of the SPTN.

5.1.1 Concatenating Transport Protocols

The MPTN gateways support end-to-end MPTN connections between MPTN access points. MPTN connections through the MPTN network are formed by concatenating a series of *MPTN segments*. An MPTN segment is a connection through the transport provider functions of two MPTN nodes across a single SPTN. This was discussed previously in 2.4, “Network Types and Connections” on page 22.

5.1.2 Routing within MPTN Networks with Gateways

Each gateway currently attaches to a native SPTN and a nonnative SPTN with respect to the transport user protocol. Figure 37 shows a typical configuration where the larger SPTN is the nonnative SPTN and the smaller ones are native SPTNs.

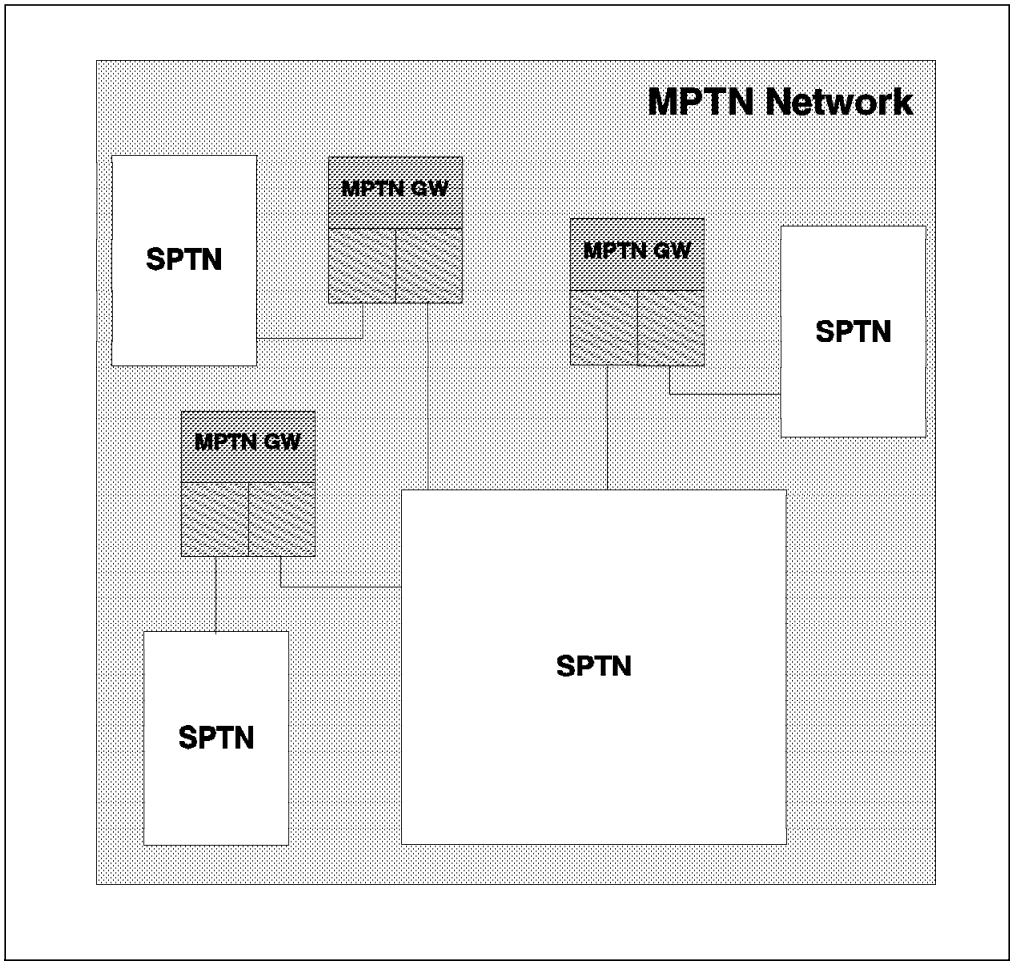


Figure 37. A Typical MPTN Gateway Configuration

At the point where the gateway receives a native connection request or a datagram, the native origin and destination transport user addresses must be converted to nonnative transport provider addresses. The MPTN gateway node behaves in the same way as an MPTN access node and has the same address

mapping services components as the MPTN access node. The same techniques, algorithmic mapping, use of a transport provider specific directory or an address mapping server, may be used as in the case of an MPTN access node.

What is particularly valuable for the MPTN gateway node is that transport user addresses may be defined in a generic manner which allows many addresses to be mapped to the transport provider address of the next, typically, final gateway leading to the SPTN to which the destination transport user node is attached. Figure 38 illustrates an MPTN gateway configuration with parallel gateways. Connections or datagram traffic may be distributed over parallel gateways again by use of generic transport user addresses.

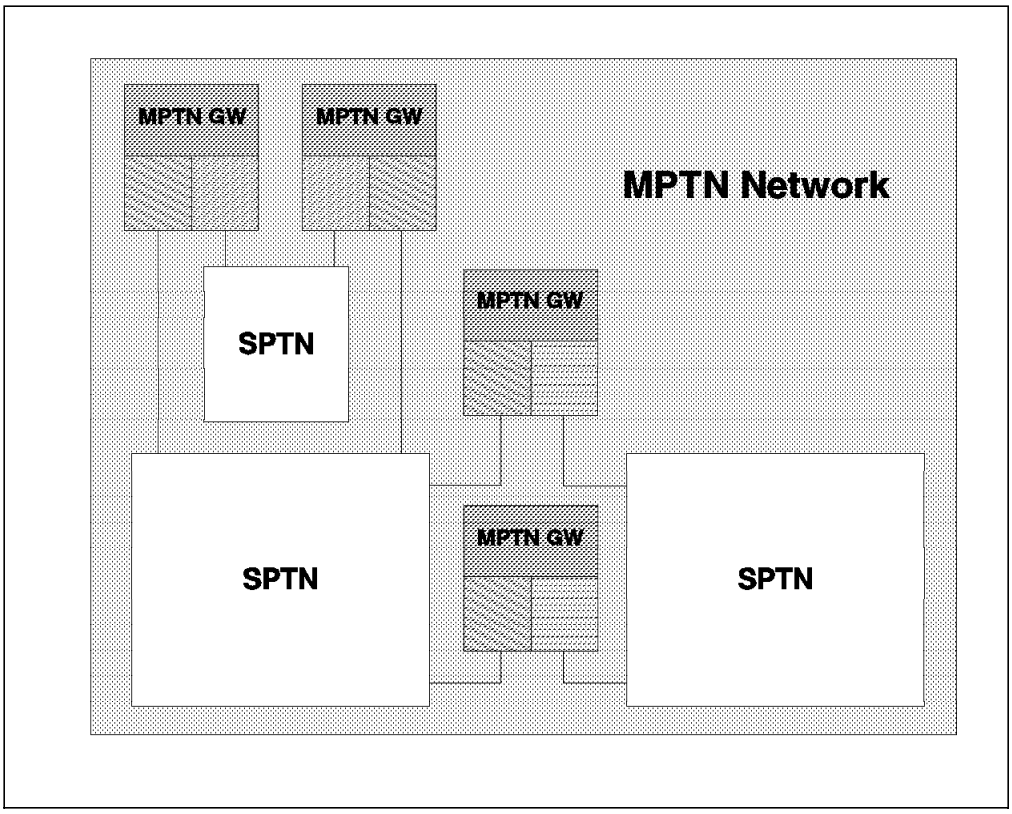


Figure 38. MPTN Network with Parallel MPTN Gateways

5.2 MPTN Gateway Node Structure

The complete structure and components of an MPTN Gateway are shown in Figure 39 on page 72. Many of the gateway components are common to the access node and hence have already been covered in Chapter 4, “MPTN Access Node” on page 53. In addition to these, there are gateway-specific components and enhancements to existing components that were not covered. We will be looking at these components here in conjunction with other elements and functions of the gateway node. The additional components are:

- Gateway services
- Network routing support
- Gateway relay

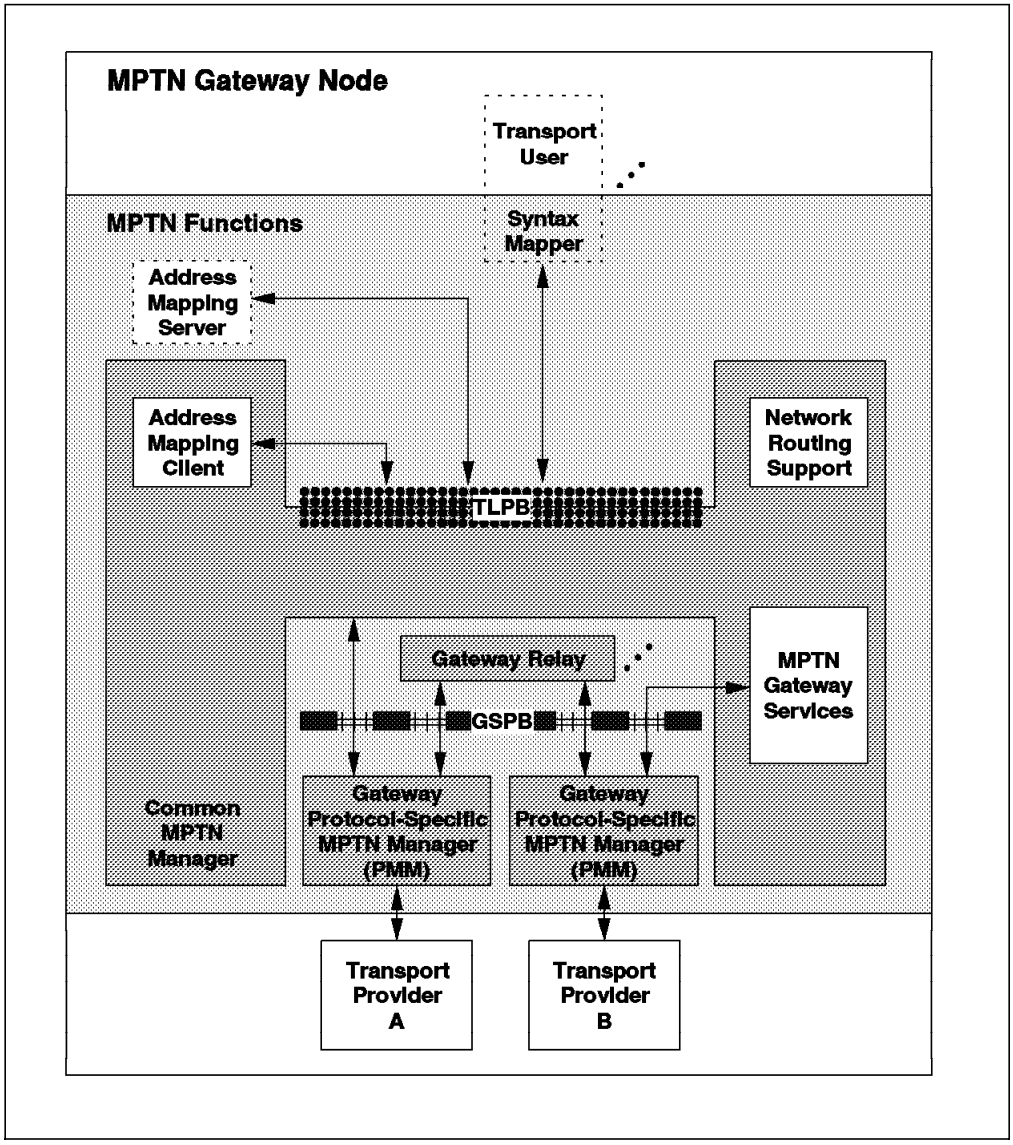


Figure 39. MPTN Gateway Structure

The structure shown in Figure 39 applies both to an MPTN gateway that supports transport users locally (provides the functions of an MPTN access node), and one that does not.

As illustrated in Figure 39 a number of components are collectively grouped under the category of common MPTN manager, CMM, components.

- **Network routing support** determines the next hop on which to forward connection requests and datagrams. The next hop could be another gateway or it could be the destination node if it is in an SPTN to which the gateway is attached. As the MPTN Connect command flows from the origin node to the destination node, each gateway maps the destination's transport user address (carried in the MPTN Connect command) to the destination transport provider address for the next segment (to the actual destination node or the next gateway).
- **Address mapping services** is responsible for the conversion of transport user addresses to one or more transport provider addresses, typically just one. Depending upon the technique employed this may involve the address

mapping client component communicating with an address mapping server in any of the SPTNs to which the MPTN gateway node is attached. This is the locate function.

- **Gateway services** assists in the establishment of MPTN connections. It creates a relay component for each connection that passes through the gateway.

In addition to the components of the common MPTN manager, the gateway supports the following components:

- A **gateway relay** component is created for each MPTN connection passing through the MPTN gateway node; it concatenates two SPTN subnetwork connections together. It receives data from the incoming SPTN connection via the GSPB and forwards the data on the correct outgoing SPTN connection via the GSPB. Once a gateway relay instance is created and an MPTN connection established, data can flow between the connected SPTNs without intervention of the gateway′s CMM.
- The **gateway protocol-specific MPTN manager** (gateway PMM) is based on the PMM, as described in Chapter 4, “MPTN Access Node” on page 53, with several (fairly small) gateway-specific extensions.

  A gateway PMM supports a specific transport protocol type and interacts with the gateway′s CMM and gateway relay components through GSPB verbs. It terminates the transport provider connections that are used to carry MPTN connections passing through the MPTN gateway. In any given MPTN gateway there will be at least one gateway PMM. The gateway PMM interacts with the transport provider it supports to take part in the transport provider′s native routing protocol.
- An **address mapping server** can optionally reside in an MPTN gateway.
- The **gateway-specific protocol boundary (GSPB)** is not a component, it defines the protocol boundary between the gateway PMMs and the gateway′s CMM, as well as between the gateway PMMs and the gateway relay components. The semantics of the GSPB are very similar to those of the BSPB, indeed the GSPB uses the verbs of the BSPB with some additions necessary to support gateway-specific functions.

5.2.1 The Gateway Protocol-Specific MPTN Manager

The structure of a gateway PMM is basically the same as the structure of a PMM in an MPTN access node, with a few gateway-specific additions. A gateway PMM performs management functions specific to a transport provider. In addition to the functions of a PMM in an MPTN access node, it is responsible for forwarding native flows, received by the transport provider, that need to be processed by the MPTN gateway. As shown in Figure 40 on page 74, the gateway PMMs and the CMM interact using the GSPB.

The *gateway protocol compensator* is similar to a protocol compensator in an MPTN access node. It compensates for required transport functions not supported by the transport provider.

The gateway protocol compensator does not have to pass complete records across the GSPB. This contrasts with a protocol compensator in an MPTN access node, which has to pass complete records if the transport user requires them. The gateway relay function supports partial records. Even though it does not have to pass complete records, the gateway protocol compensator must still

keep track of record boundaries and indicate when complete records have been passed to the gateway relay.

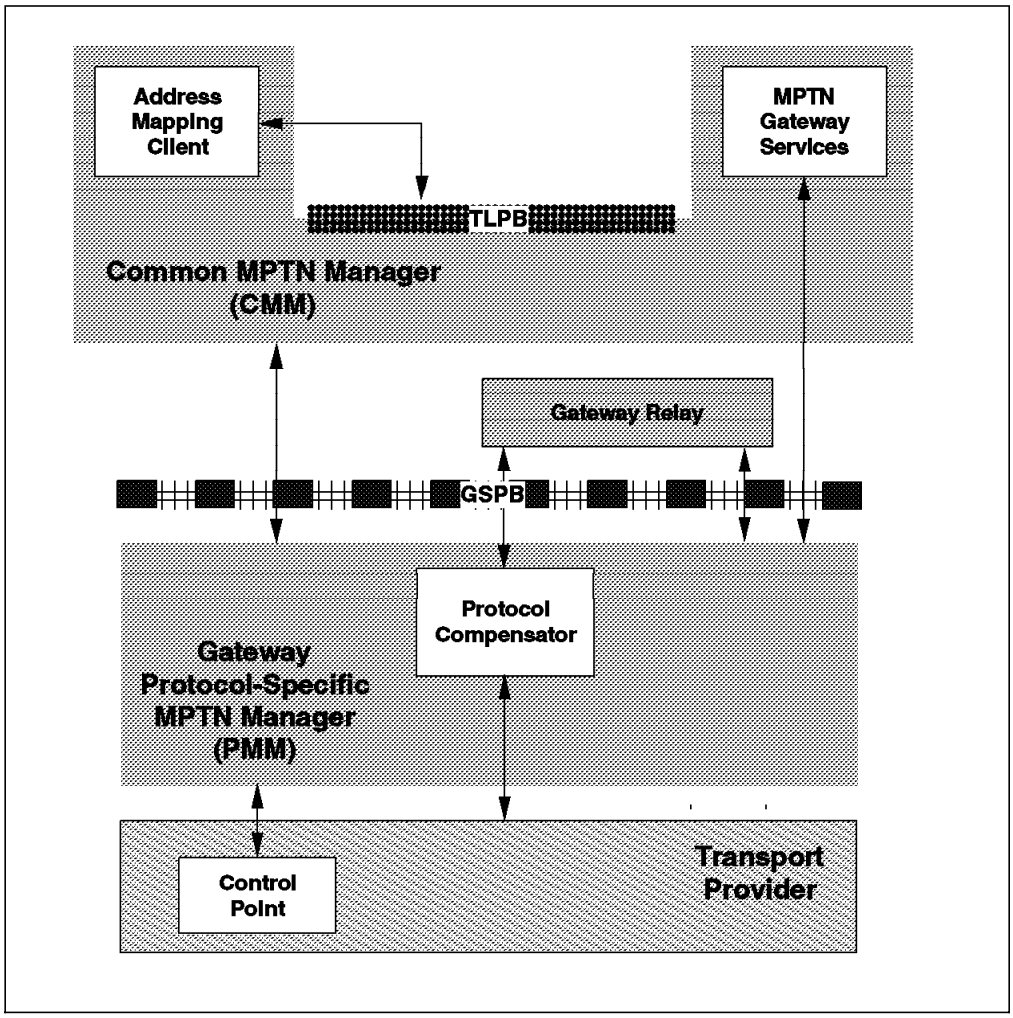


Figure 40. Gateway PMM Components

The gateway PMM also maps the GSPB verbs to primitives that are expected by the transport provider.

5.3 Protocols between End Nodes and MPTN Gateways

A native end node is a node not containing any MPTN functions. As such the transport user and the transport provider belong to the same protocol family.

An MPTN gateway can be accessed by a native end node using the native protocols of the network, or by an MPTN access node using MPTN protocols. Thus transport users in both native nodes and MPTN access nodes can interoperate with their peers across the MPTN network. For example, an application written to the AF_INET sockets interface, supported using the MPTN architecture on an SNA network, can interoperate with a peer application written to the AF_INET sockets interface on a native TCP/IP network.

5.3.1 Native Access

Native protocols are used between a native end node and an MPTN gateway. In this case the MPTN gateway must participate in the native routing protocols of the network in order to receive connection requests and datagrams that are to be forwarded through the MPTN network.

There are two fundamental types of routing protocols with which MPTN gateways can interact to provide native access:

- **Routing-table-based protocols** continuously distribute reachability information, which includes the path, and characteristics of that path (e.g., delay, cost, security), to a destination or set of destination end systems. When a route to a particular destination is required, it must already be known at each routing node. Otherwise the destination is assumed to be unreachable.
- **Search-based protocols** only perform routing functions when a path to a particular destination is required. Then they broadcast a search for the required destination in order to determine how to route to that address.

For *routing-table-based protocols*, the MPTN transport gateway must be able to exchange reachability information with the native protocol. That is, it must be able to advertise within the native protocol reachability to end systems outside of the native domain, and it must be able to learn what systems can be reached in the attached SPTN. With this capability, the native protocol will be able correctly to route connection requests and datagrams destined for nodes outside its SPTN to the gateway, and the MPTN network will be able to route data to the native domain.

For *search-based routing protocols*, the MPTN transport gateway must be able to receive all searches and respond when the particular resource can be reached across the MPTN network. It sends a positive reply to the search if it can reach the address, which will cause subsequent connections or datagrams to that address to be routed to the MPTN transport gateway by the native protocols.

Once the native routing protocols have determined that a connection or datagram is to be routed to an MPTN gateway, the native protocols are used to establish the connection or forward the datagram. Similarly, native protocols are used in the native access case to forward data from an MPTN transport gateway to a destination end system.

5.3.2 Nonnative Access

Whenever native protocols cannot be used, MPTN address mapping and signaling protocols are used between MPTN access nodes and gateway nodes. Gateways register all the transport users that can be reached through it. Wildcards may be registered. If an access node sends a connection request or a datagram, the gateway will receive it and forward it to the actual destination.

5.4 Protocols Between MPTN Gateways

5.4.1 Connection Management Protocols

Once an MPTN connection has been established, the main MPTN gateway function is to perform the relaying of data packets between the corresponding pair of connection segments. Although conceptually simple, this function faces the possible hazard of internal buffer depletion due to differences in throughput between the joined segments. The MPTN architecture solves this problem by providing buffer usage limits for each connection. If data cannot be sent on the outgoing connection, no buffers are freed and the relay stops receiving data, thereby applying back pressure to the incoming connection. Once the back pressure is applied, the native protocol will exercise its flow-control procedures on the connection segment, and hence the back pressure propagates hop-by-hop back to the source, which reduces its input to a sustainable rate.

When packets may have to be segmented, the reassembly function is placed in the access nodes, or destination transport gateways for native access, so that repetitive reassembly is avoided.

Connection termination follows a procedure similar to that for connection establishment, with the connection being terminated hop-by-hop until it is closed end-to-end.

5.4.2 Datagram Protocols

The procedure for forwarding datagrams parallels that of sending data packets on a connection.

Chapter 6. MPTN Address Mapping Services

Each transport provider must support an addressing scheme to allow users to establish communications with specified partners. The terms *address* and *name* are widely used in communications protocols to identify the location of an entity. In MPTN, the term address is used. Different transport protocols exist in an MPTN network, so there are different address spaces. Addresses tends to be structured in a form such as (netid.hostid.local-address) where:

netid is a name given to a subnetwork, a part of the network as a whole, which consists of a set of hosts with a logical association between them

hostid is a name given to an individual host within a subnetwork

local-address is a name given to a specific process on an individual host

When different protocols are used above and below the TLPB, there will be different address spaces involved. Therefore, MPTN provides a mechanism, called *MPTN address mapping*, that allows each transport user to:

- Register its transport-user address so that it is accessible to users of the address mapping services
- Map a destination transport-user address to an address understood by the serving transport provider

In the following sections we will see how MPTN:

- Keeps the addresses unique all over the whole network so that the addresses are accessible from anywhere
- Maps the transport-user addresses to addresses of the underlying transport provider SPTNs
- Finds addresses, so that a route from the sender to the receiver is found

6.1 MPTN Addresses

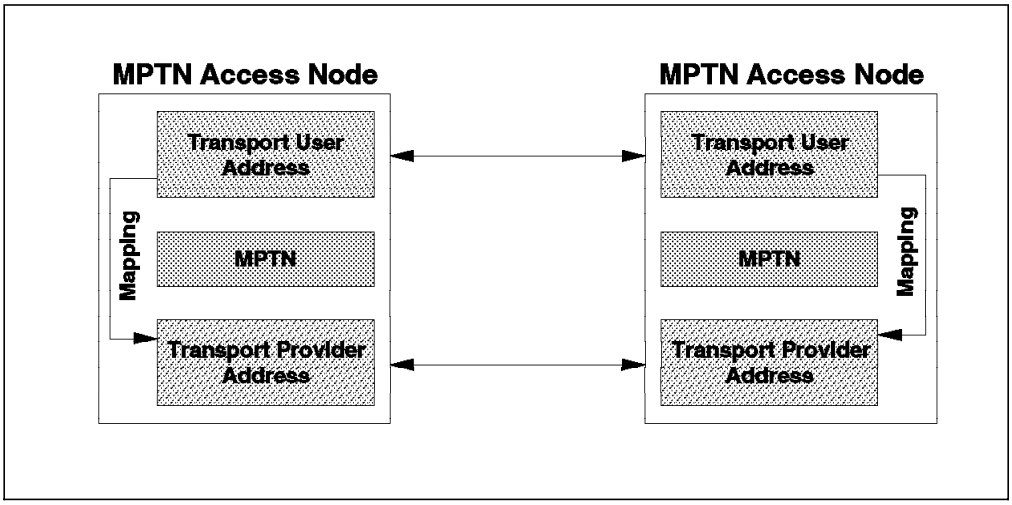


Figure 41. MPTN Address Space Mapping

In order to allow unambiguous identification of communicating partners, transport addresses must be unique. In SNA, OSI, and TCP/IP, transport address uniqueness is controlled by administration, while in NetBIOS a protocol is used to detect duplicate addresses (names). Within MPTN the network addresses also have to be unique. In the case of an MPTN access node, two address spaces need to be considered as shown in Figure 41 on page 77:

- Transport-user address space (above TLPB)
- Transport-provider address space (below TLPB)

If the address spaces are identical, the transport-user addresses are native to the transport network; otherwise the transport-user addresses are nonnative.

Addresses used within each protocol must be unique. However, since addresses are simply bit strings, the same address may appear in different protocol address spaces. Thus MPTN qualify addresses using an identification of the protocol managing the address space.

6.1.1 MPTN Address Format

To allow duplicate addresses to be used in different protocol address spaces, MPTN uses an ordered pair (address type, protocol-specific address) to make transport-user and transport-provider addresses unique in the MPTN network. Such an ordered pair is called an *MPTN-qualified address*. The address type indicates the format of the protocol-specific address.

For a detailed discussion of address types and the address formats used by the different communication protocols see 2.5, “Address Formats and Mapping” on page 25.

6.1.2 Well-Known Addresses

Well-known addresses are centrally-administered or architecturally-defined local addresses. Using well-known local addresses significantly simplifies the MPTN address mapping services. PMM uses the well-known local address to identify the PMM function for the relevant transport provider in an MPTN node. In this case the service requested will be the MPTN nonnative service. Other local addresses will apply to native services.

For each transport protocol this well-known local address must be defined either at installation time or by default.

TCP a well-known port

UDP a well-known port

SNA(TP) a reserved transaction program name (TPN)

OSI a reserved Tselector

NetBIOS generated NSAP

For example, MPTN uses port number 397 for TCP/IP. In the current implementation of sockets over SNA, the service TPN is defined as X¢28F0F0F1¢.

As the required local addresses are well known, MPTN needs to maintain only the node addresses within its addressing scheme.

6.2 MPTN Address Mapping Services Model

The following will discuss the two major address mapping services functions which are used to map the transport-user address to the transport provider address dynamically. These two functions provide for:

- Address registration, by which a transport user makes its address available to other transport users
- Locating a partner for a transport user, allowing the transport user to set up a connection with the partner or to send a datagram to the partner

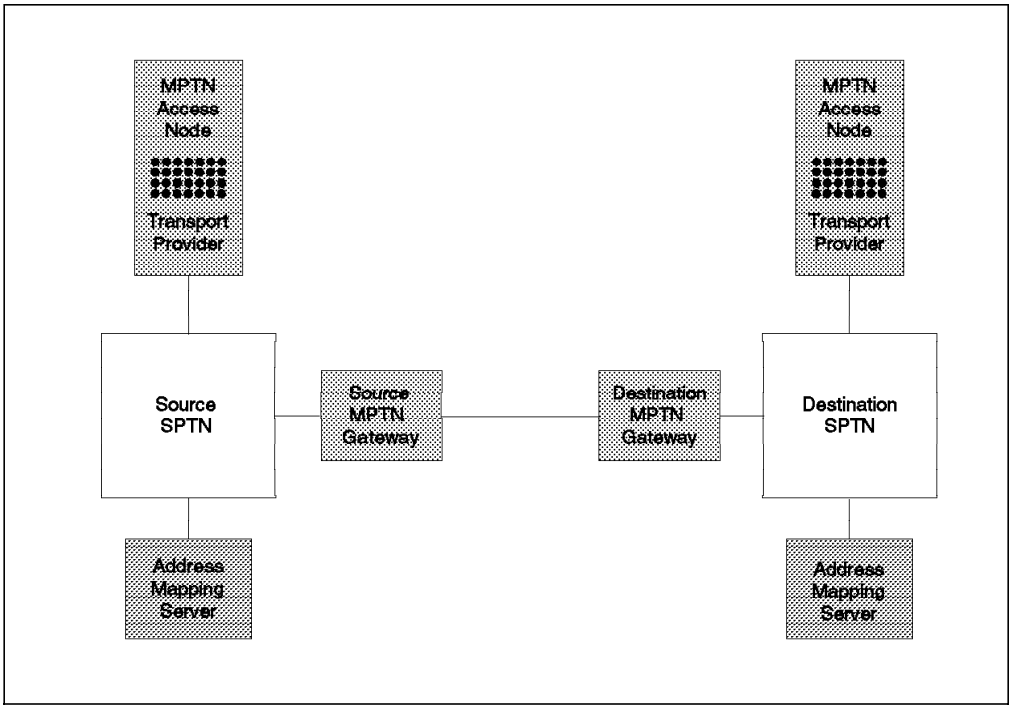


Figure 42. Address Mapping Services Overview

These functions use the following three basic components of MPTN, as shown in Figure 42:

- MPTN access node, from which transport users request services
- Address mapping services, by which a transport-user address and a transport-provider address are bound to each other
- The gateway, through which the internetwork connectivity between transport-user addresses is made possible

The address services mechanism must provide every MPTN access node and gateway node with the same mapping of transport-user to transport-provider addresses.

6.2.1 Registering Addresses

In the MPTN network, individual transport-user addresses are registered. An exception is multicasting where group addresses are registered.

6.2.1.1 Registering Individual Addresses

When a transport user initializes its interface to MPTN it has to specify which address type it uses and the transport-user address by which it is known to the network. When specifying its transport-user address (using an M_BIND_DC verb) it can request that its address be registered with an external directory server or MPTN address mapping service to make it known within the entire MPTN network.

The address mapping services of the CMM, called the *address mapping client*, register the transport-user address in the MPTN format (address type, protocol-specific address) together with the corresponding transport-provider address or addresses, also in the MPTN format. The reason that the address mapping client may register more than one address is that there might be more than one transport provider installed in the access node or gateway node. This transport-user address has to be mapped to all nonnative transport providers available. In the MPTN network only the address type and node address of the transport user address or transport-provider address are reported to the address services mechanisms.

6.2.1.2 Registering Group Addresses

A transport user may also join a multicast group by specifying a group address when requesting registration of a transport-user address. The group address registration determines the MPTN multicasting model. It is essential to use a consistent transport-provider group address for a particular transport-user group address, so that multicasting to the transport user′s group address can be done by using the single transport provider′s group address.

The access node itself does not supply a transport-provider group address for a transport-user group address. The access node sends the registration request to address mapping services. If the transport user′s group address already exists, address mapping services know the corresponding transport-provider group address already and returns it. If it does not exist, address mapping services acquires a corresponding transport-provider group address, registers the group address, and returns the transport-provider group address. A further consideration with multicasting is whether the transport provider supports multicast transmission of datagrams. Where the transport provider does not, MPTN defines the use of a multicast server.

6.2.2 Address Resolution - Locating a Partner

Address resolution maps a transport-user address to one or several transport-provider addresses. It is required when a transport user requests a connection or sends a datagram to a destination address that cannot be reached by means of a native transport provider in the MPTN access node.

The address mapping client component in the access node or gateway node is invoked to send the destination transport-user address to the address services mechanism for address resolution.

6.3 Architecture of Address Mapping Mechanism

The previous overview section has shown that all parts (the MPTN access node, the MPTN gateway, and the address mapping services) are involved if full connectivity within the MPTN is required.

The following section looks at three possible implementations for the address mapping services mechanism:

1. MPTN address mapping service
2. Algorithmic mapping
3. Extended native directory

An SPTN may support more than one implementation for its address mapping services mechanism. The following sections describe how each implementation provides the two major service functions, and their limitations.

6.3.1 MPTN Address Mapping Service

The MPTN address mapping service consists of two logical components:

- An *address mapping client* component residing in an MPTN access or gateway node
- An *address mapping server* component residing in any MPTN node, including a node dedicated to providing the access mapping server function

Both the address mapping client component and the address mapping server component are *direct transport users* of the TLPB (see Figure 30 on page 56). The address mapping client or server components may reside with other transport users in an MPTN node, access or gateway.

The address mapping server maintains a table which defines the transport user addresses and their mapping to one or more transport provider addresses.

The address mapping clients register these addresses with the address mapping server whenever an access node is initialized or an address mapping server declares it is active and indicates that it does not have, or is discovered not to have, the correct set of mappings for the node where the address mapping client component resides. Figure 43 on page 82 shows the structure of the address mapping service.

The address mapping server needs to communicate with the address mapping client in MPTN access nodes and MPTN gateway nodes. As we have seen above, an MPTN access or gateway node needs connectivity to an address mapping server if a node address must be found to establish a connection or send a datagram to a partner. An MPTN node does not need to use an address mapping server if it uses one of the other techniques, algorithmic mapping or an extended native directory, for its address mapping services mechanism. However, if an address mapping server is used, it must be accessible from all MPTN nodes, access and gateway, on the nonnative SPTN to which the MPTN access and gateway nodes attach.

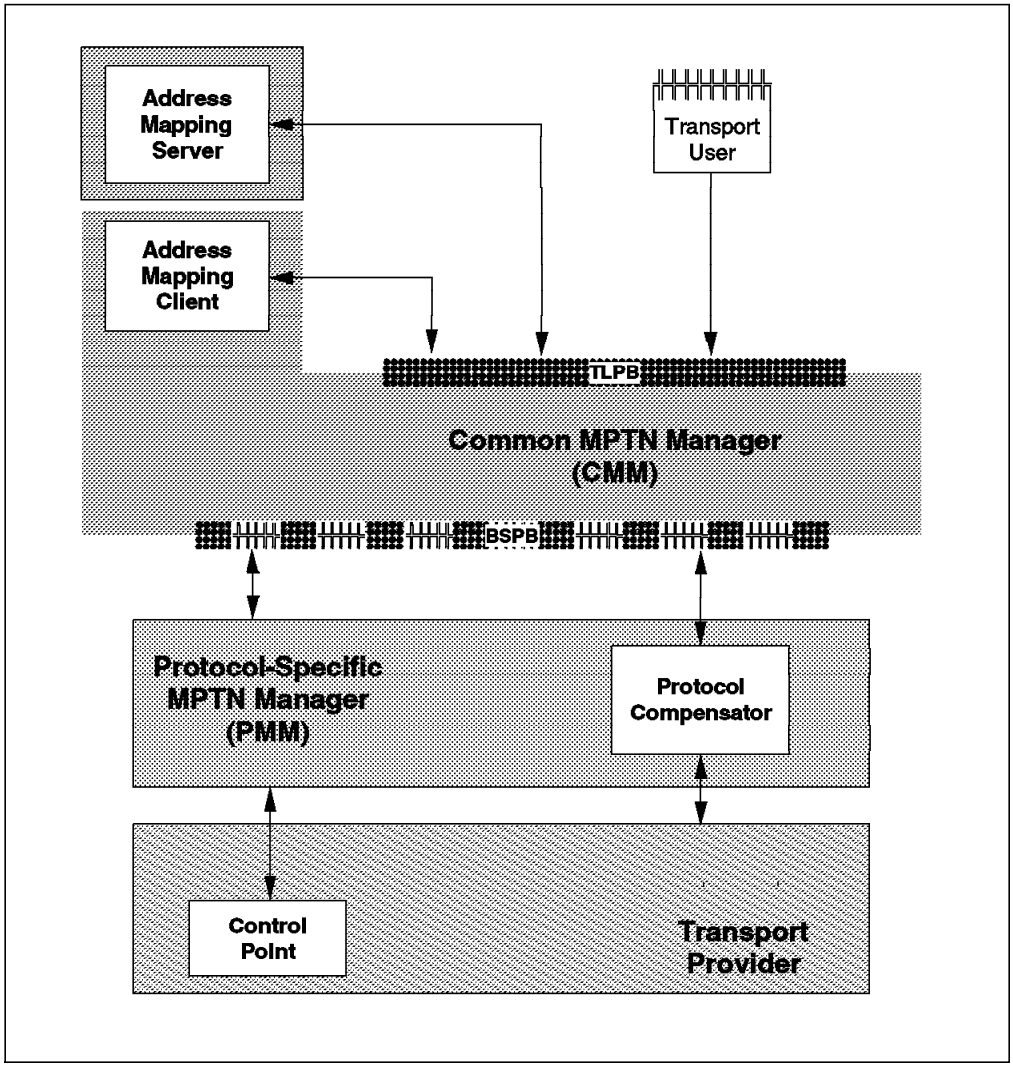


Figure 43. Address Mapping Service

6.3.1.1 MPTN Address Mapping Service Details

The address mapping client and address mapping server communicate according to a set of protocols defined by the MPTN address mapping service.

Since the volume of traffic is generally small and intermittent and the TLPB necessarily provides mechanisms for connectionless transport, connectionless transport has been chosen as the communications mechanism. In addition, there could be a large number of simultaneous relationships between an address mapping server and all the address mapping clients which employ its services. If connections were to be established, this could consume excessive resources on an MPTN node supporting the address mapping server component.

However, connectionless transport is unreliable in that messages may be lost or discarded. This means that there must be an alternating flow of a message sent, confirmed by a message received, by the communications partner responsible for assuring that a transaction is performed. If the confirmation is not received within a certain period of time, the transaction must be repeated, up to a retry limit, until the confirmation message is received. Since transactions may be repeated, there is a possibility that duplicate messages are received and care is

needed that repeating a transaction does not distort the information recorded as a result of receiving duplicate messages.

The MPTN address mapping service uses a default timeout of 30 seconds or 2 minutes (see later) for receiving a confirmation message. The default retry count is 5. An address mapping transaction is either a request to register a transport user address with all its associated transport provider addresses or a request for address resolution. Thus the transactions are described as *idempotent*, meaning they are not affected by duplicates.

The functions defined by the address mapping service are as follows:

- Address mapping client initialization

  This allows an address mapping client to identify an address mapping server by means of the transport provider address of the address mapping server. The address may be found by a protocol specific to the SPTN or it may be by means of definition in the address mapping client.

  The final steps of initialization are to register (see later) the address mapping client transport provider address or addresses.

- Address mapping server initialization

  An address mapping server maintains a list of address mapping clients and their transport provider addresses in nonvolatile storage. When an address mapping server initializes, it notifies each known address mapping client. With this notification is an indication whether or not the previous registrations have been preserved or not. If the indication is that previous registrations have been preserved, there is a count of how many such registrations entries there are so that the address mapping client can test, just on the basis of the number, whether the address mapping server has the correct information. If previous registrations have not been preserved or the count is incorrect, the address mapping client will initiate clearing and registering again as necessary.

- Address mapping server to client path integrity

  For each address mapping client noted in nonvolatile storage the address mapping server maintains a time stamp recording the time of the last communication with the address mapping client.

  As part of the address resolution function (see later), the address mapping server supplies a mapping of transport user to transport provider addresses to any address mapping client making such a mapping request. If subsequently the MPTN node discovers that the transport user address is “unreachable,” the associated address mapping client may report the problem to the address mapping server. After checking whether the address mapping client responds to requests, the address mapping server will then mark the registration entry for the transport user address, which has an associated address mapping client entry, as “of dubious value.” This “of dubious value” qualification can then be returned on any subsequent locate request which includes a transport provider address corresponding to this transport user address.

  Note: If the address mapping client is operating correctly, it should register all transport user addresses when the function associated with the transport user address is started and deregister the transport user address when the function associated with the transport user address is stopped or abnormally terminates. Delays in operating these deregistrations may cause

inconsistencies over a short period of time which could lead to “of dubious value” being set briefly quite normally. The only case remaining where “of dubious value” markers are *not* self-correcting is when there has been a node failure and the associated address mapping client has failed also.

When a transport user address has been reported as “unreachable,” the address mapping server sends a datagram to the associated address mapping client to try to solicit a response. If a response is received, it is assumed that the address mapping client saw no reason to deregister the transport user address which was has been described as “unreachable” and so the “of dubious value” marker is not set. It is assumed that some unknowable fault cause the failure in the MPTN path. If a response is not received, the “of dubious value” marker is set in the entry of the affected transport user. Every two hours, the address mapping server will check the time-stamp of each served address mapping client which has associated registration entries marked “of dubious value.” If the time stamp indicates that such an address mapping client has had no communication for two hours, all registration entries for that address mapping client, and the entry for the address mapping client in nonvolatile storage, are removed.

- Address registration

  When a new transport user with its associated address becomes active in a node, the address mapping client component of the CMM will register the transport user address together with its one or more transport provider addresses with the address mapping server. This enables the address mapping server to provide the mapping between the transport user address and the one or more transport provider addresses to any other address mapping client which uses an address resolution request for that transport user address.

  When the transport user address no longer corresponds to an active transport user in the node, the address mapping client should deregister the transport user address with the address mapping server.

- Address resolution

  When the address mapping client component of a CMM receives a request to provide the transport provider address for a destination transport user address, the address mapping client sends a locate request to an address mapping server. The address mapping server returns the one or more transport provider addresses corresponding to the supplied transport user address, if the mapping is available. Otherwise, a negative response is returned indicating no mapping.

  If subsequently the transport user appears to be “unreachable,” the address mapping client informs the address mapping server which will mark the transport user entry “of dubious value” and will attempt to contact the associated address mapping client in order to verify whether the MPTN node has failed.

  In order to support potential management functions associated with MPTN, a reverse mapping is available on request from the address mapping server such that the address mapping server will return all transport user addresses associated with a supplied transport provider address.

When an address mapping client or server issues a *request* it expects a confirmation message within the short-timeout period of 30 seconds. Normally, the address mapping server will be able to send a reply within these 30 seconds.

There may be instances, however, when a final reply cannot be sent within these 30 seconds, for example, because the local server has to query some other address mapping server. In this case, a *response* is sent in order that the timeout does not expire, that indicates that the transaction is pending. A pending response indicates also that, subsequently, there will be a reply which contains the requested information. In order to complete the transaction to the satisfaction of both partners, the reply must be confirmed with another response. Since pending indicates that the reply will follow after some time, a longer timeout of 2 minutes is applied to the receipt of the reply.

A further feature of the protocol is that each transaction has an unique correlator set by the originator of a transaction which is specified in each request, response and reply message and there is no requirement that one transaction be complete before another can be initiated.

The behavior of an address mapping server when an address mapping client fails to respond was described before. The behavior of an address mapping client when an address mapping server fails can be one of the following:

- Stops using an address mapping server, possibly relying only on address mappings which have been cached
- Saves any requests that appear from activity within the node and try to contact the address mapping server later
- Attempts to locate another server and synchronize with it

  In this case, a new server may be found on the same SPTN as the original server because one of the MPTN nodes, detecting that the usual address mapping server had failed, assumed the role of an address mapping server. An alternative is that two address mapping servers were active on the SPTN. It may even be that the original server could be contacted over another SPTN. The newly discovered server could have been sharing the table of address entries that was used by the original address mapping server so that reregistration by the address mapping clients will not be necessary.

6.3.1.2 Address Mapping Server Configurations

The following are some samples to illustrate possible address mapping server configurations:

1. Several transport providers may share a common physical medium. All transport providers of the same type comprise an SPTN. Thus, multiple SPTNs can coexist on the same physical medium. All of these SPTNs can share one address mapping server. For example, TCP/IP and SNA transport providers may be used on a token-ring LAN. In this case, only one address mapping server is required which can be reached using TCP/IP or SNA. Therefore, an MPTN access node needs to use only one transport provider for communication to the address mapping server, even if several transport providers are installed at the MPTN access node, and may report in a single flow addresses for all its installed transport providers when it registers an address. This is shown in Figure 44 on page 86. (The figure shows two connections to the LAN, but there may be only one LAN adapter actually installed which is shared by both transport provider protocols.)

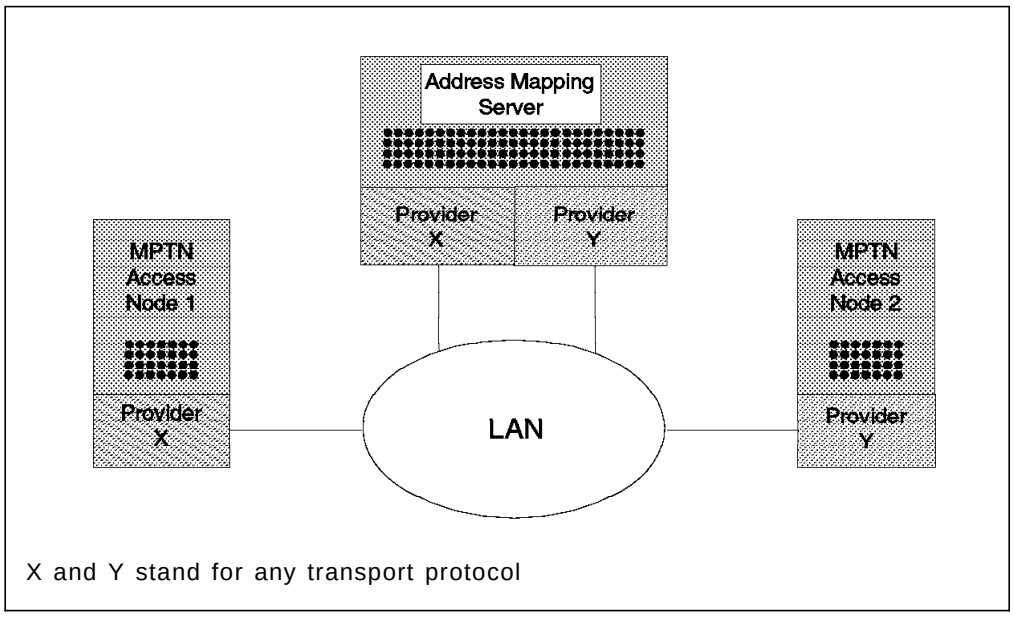


Figure 44. Address Mapping Server Sharing by Multiple SPTNs

2. An address mapping server may also reside in an MPTN node which serves the attached MPTN subnetworks, as shown in Figure 45.

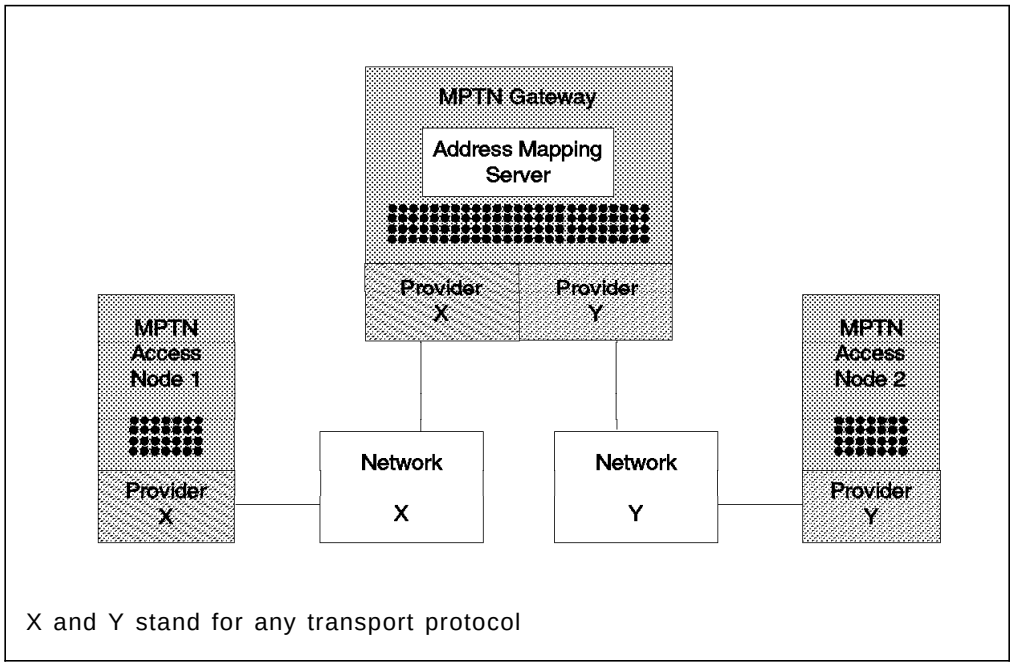


Figure 45. Address Mapping Server Serving Multiple SPTNs

3. An access node may also attach to several SPTNs and thus be connected to multiple address mapping servers if these subnetworks do not share the same address mapping server, as shown in Figure 46 on page 87.

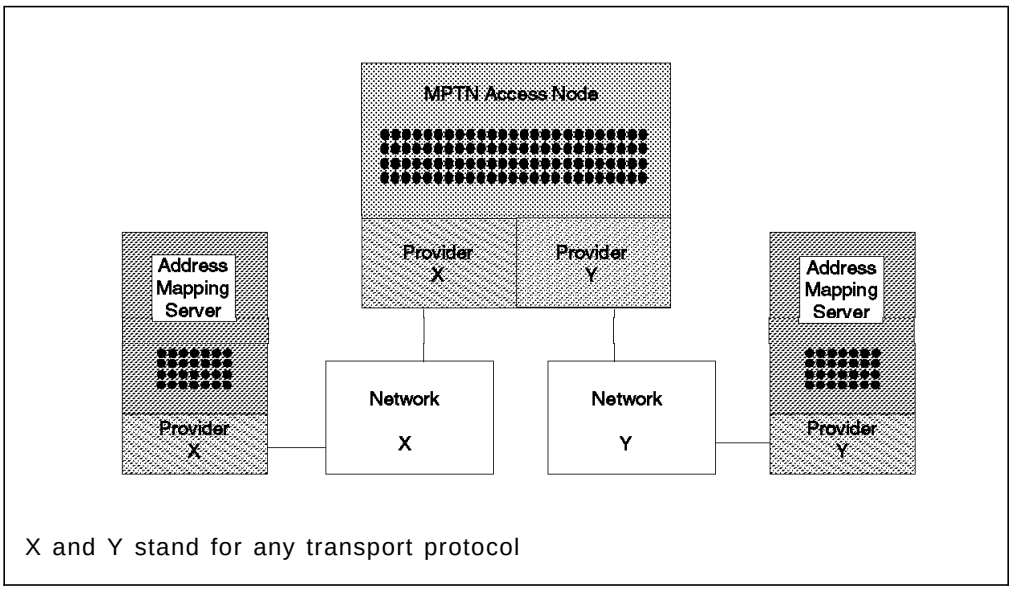


Figure 46. Access Node Attached to Multiple SPTNs

6.3.2 Algorithmic Mapping of Addresses

With algorithmic address mapping, the address mapping services applies an algorithm to the transport provider's address A which results in a corresponding address B which is used across the transport provider's network. This is illustrated in Figure 47.

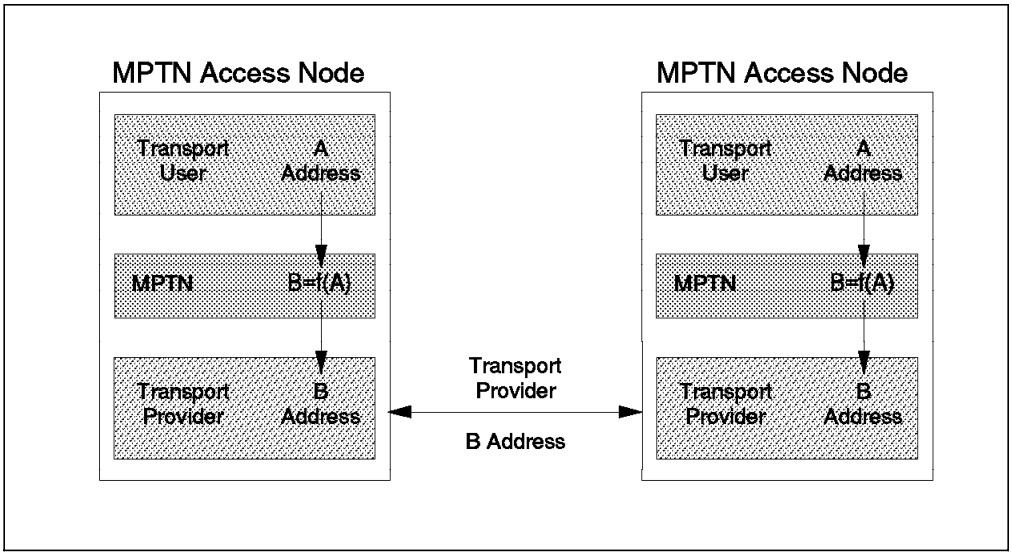


Figure 47. Address Mapping: Algorithmic

The new transport-provider address is registered in the existing directory of the transport provider's protocol. This method is appropriate when the user address space can map into the provider's address space. The transport provider's address space needs to be larger than the transport user's address space.

An example of address mapping is applying an algorithm to an internet address to obtain an SNA name when a TCP/IP application wants to communicate across an SNA network. See the AnyNet product publications for the *Sockets over SNA*

functions for descriptions of the algorithm implemented in the AnyNet product family to map internet addresses to SNA LU names.

Every MPTN access node and MPTN gateway uses the same algorithm for a given combination of transport user and transport provider. There is no table required to hold the mappings. When transport-user addresses are registered, they are converted to the corresponding transport-provider addresses which may be registered to the corresponding transport provider's native directory. (In the case of the IP *domain name server*, this is *not* possible.)

6.3.3 Extended Native Directory

Another alternative is to enhance existing protocol-specific directories to handle addresses of different formats. With this alternative, all transport-user addresses can be registered in the native directory of the transport provider. For example, the TCP/IP domain name server could be used.

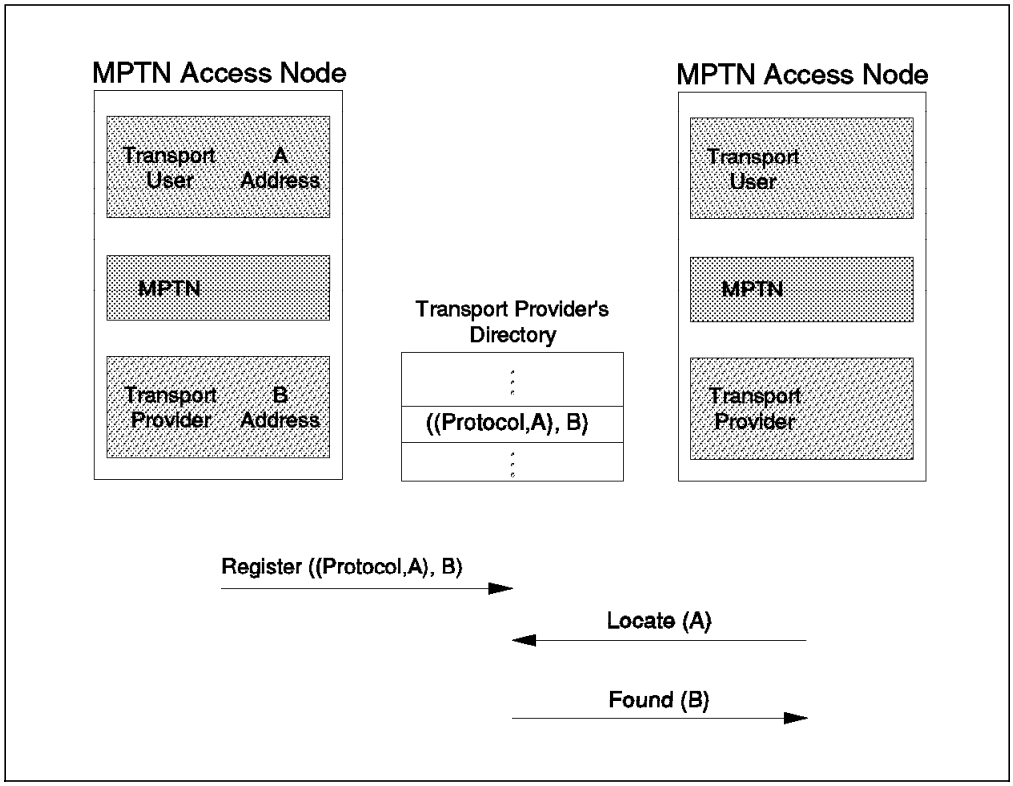


Figure 48. Address Mapping: Extended Native Directory

A transport provider's native directory can be used only if it supports all functions that are requested by a transport user (and normally supported by its native directory services) when registering addresses.

In the example, the TCP/IP domain name server can be used to support SNA names. When TCP/IP is the transport provider, the SNA transport-user address NETA.LU1 must be defined as LU1.NETA.SNA.IBM.COM in the domain name server. Note that the transport-user address must be defined rather than a registration protocol being used because, in the case of the TCP/IP domain name server, there is no protocol for dynamic registration.

6.3.3.1 Registering a Transport User Address

The transport provider may use its enhanced native directory to register the transport-user address.

6.3.3.2 Locating a Partner

When a transport user requests a connection or datagram delivery, its partner's address must be resolved. This partner address is found in the native extended directory only if it is registered or defined within this SPTN. The transport-provider address can be extracted from this directory and a connection request or a datagram can be sent to the partner in the same SPTN.

For example, in Figure 48 on page 88 when transport user A is registered in the transport provider's directory, both a protocol identifier and the user address A are registered along with the association with transport-provider address B. When a transport user requests a connection or datagram delivery, a locate is sent to the transport provider's native directory to find the destination address.

This method is appropriate when the transport provider's directory entries can be manipulated to support the registration of different address types.

6.3.4 Summary

Figure 49 on page 90 illustrates the three address mapping mechanisms we have discussed in this section.

The first part of the example illustrates the *algorithmic* method whereby a TCP/IP address can be mapped into an SNA network-qualified name. This is a two-step process. First, by using a mask, the TCP net or subnet ID is determined and then mapped to an SNA network name by means of table lookup. Next, the remaining portion of the TCP/IP address is the host ID and an algorithm is performed on this to generate the LU name.

The second part of the example illustrates the method used by an *extended native directory* to generate the TCP/IP address. The TCP/IP domain name server (DNS) is extended to store the SNA name in a form acceptable to the DNS. The convention, in accordance with the DNS convention that the least significant part of the name appears first, is to reverse the SNA fully-qualified name *NETID.LU* as *LU.NETID* and then append a suitable *domain name*, *SNA.IBM.COM* being the default. Thus, by reference to the DNS, an SNA fully-qualified name is converted to the IP address of the MPTN access or gateway node that may be used to reach the SNA LU. A similar reference to the DNS with an IP address enables the SNA fully-qualified name to be retrieved.

The third part of the example illustrates the method used by the *MPTN address mapping service*. In this case, the transport user and transport provider association is registered in the address mapping server. This occurs dynamically each time a transport user registers a transport-user address, causing a (user, provider) address pair to flow to the address mapping server for registration. Thus, when the NetBIOS name is presented to the MPTN address mapping server for resolution, the associated SNA name, *NETID.LU*, is returned.

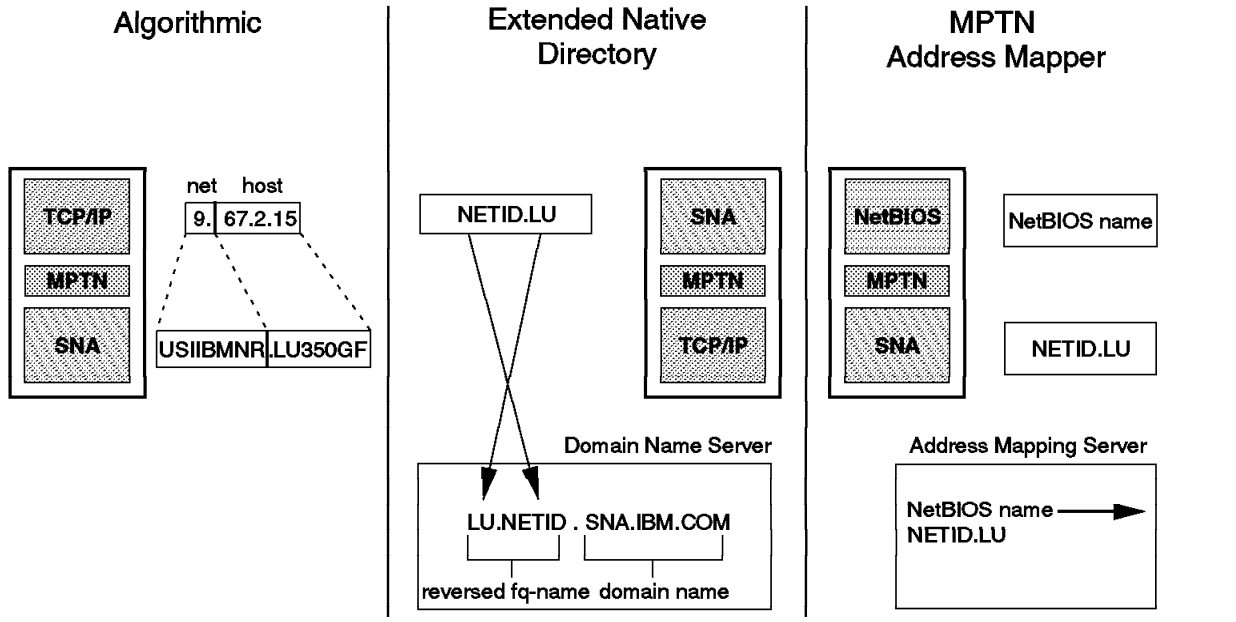


Figure 49. Address Mapping Mechanisms - Examples

The only address services mechanism which is suitable for the complete MPTN architecture is the MPTN address mapping service. Other address services mechanisms have limitations and are usually restricted to a particular transport user and transport provider pair.

Chapter 7. MPTN Product Implementations

The IBM products which implement the MPTN architecture are collectively referred to as the AnyNet family of products.

Table 4 lists the MPTN access node implementations supported by available or announced products of the IBM AnyNet product family.

Table 4. AnyNet Product Platforms - Access Nodes

Transport User	Transport Provider	MVS	AIX	OS/400	Windows	OS/2
SNA	TCP/UDP/IP	yes	yes	yes	yes	yes
AF_INET sockets	SNA	yes	yes	yes	yes	yes
NetBEUI	SNA	-	-	-	-	yes
AF_INET sockets	NetBIOS	-	-	-	-	yes
AF_INET sockets	IPX	-	-	yes	-	yes
SNA	IPX	-	-	yes	-	yes

Table 5 lists the MPTN gateway node implementations supported by avalable or announced products in the IBM AnyNet product family.

Table 5. AnyNet Product Platforms - Gateway Nodes

Transport User	Transport Provider	MVS	AIX	OS/400	Windows	OS/2	2217
SNA	TCP/UDP/IP	yes	-	-	-	yes	yes
AF_INET sockets	SNA	-	-	-	-	yes	yes
NetBIOS	SNA APPC	-	-	-	-	yes 1	yes 1
IPX	SNA	-	-	-	-	yes 1	yes 1

1 These gateway implementations are provided by the LTLW program which operates protocols other than those defined by MPTN architecture, having been developed before MPTN architecture.

Table 6 describes the supported combinations of transport user over transport provider.

Table 6. Supported Transport User and Transport Provider Combinations

Transport User	Transport Provider	Access Node	Gateway Node
SNA 1	TCP/UDP/IP	yes	yes
AF_INET sockets	SNA	yes	yes
NetBEUI	SNA	yes	yes 2
AF_INET sockets	NetBIOS	yes	-
AF_INET sockets	IPX	yes	-
SNA	IPX	yes	-
IPX	SNA	-	yes 2

Products are announced or available on the OS/2 platform for all functions.

1 LU Types other than APPC (LU 6.2) are also supported.

2 These gateway implementations are provided by the LTLW program which operates protocols other than those defined by MPTN architecture, having been developed before MPTN architecture. Thus, the access node implementation and the gateway node implementation do **not** operate compatible protocols.

The lines and arrows superimposed on the Networking Blueprint symbol in Figure 50 show sockets application transport over SNA networks and APPC application transport over TCP/IP networks. True to MPTN objectives, the application is running over a network transport other than the one usually associated with the application.

This chapter describes a product family, the AnyNet family, implementing the MPTN architecture. A key objective of MPTN is to separate the application layer of a network from the transport layer; thus, applications will not be required to be aware of the network transport protocol used to carry the application data.

This chapter especially provides details concerning the VTAM and OS/2 product delivery of the functions shown in Figure 50.

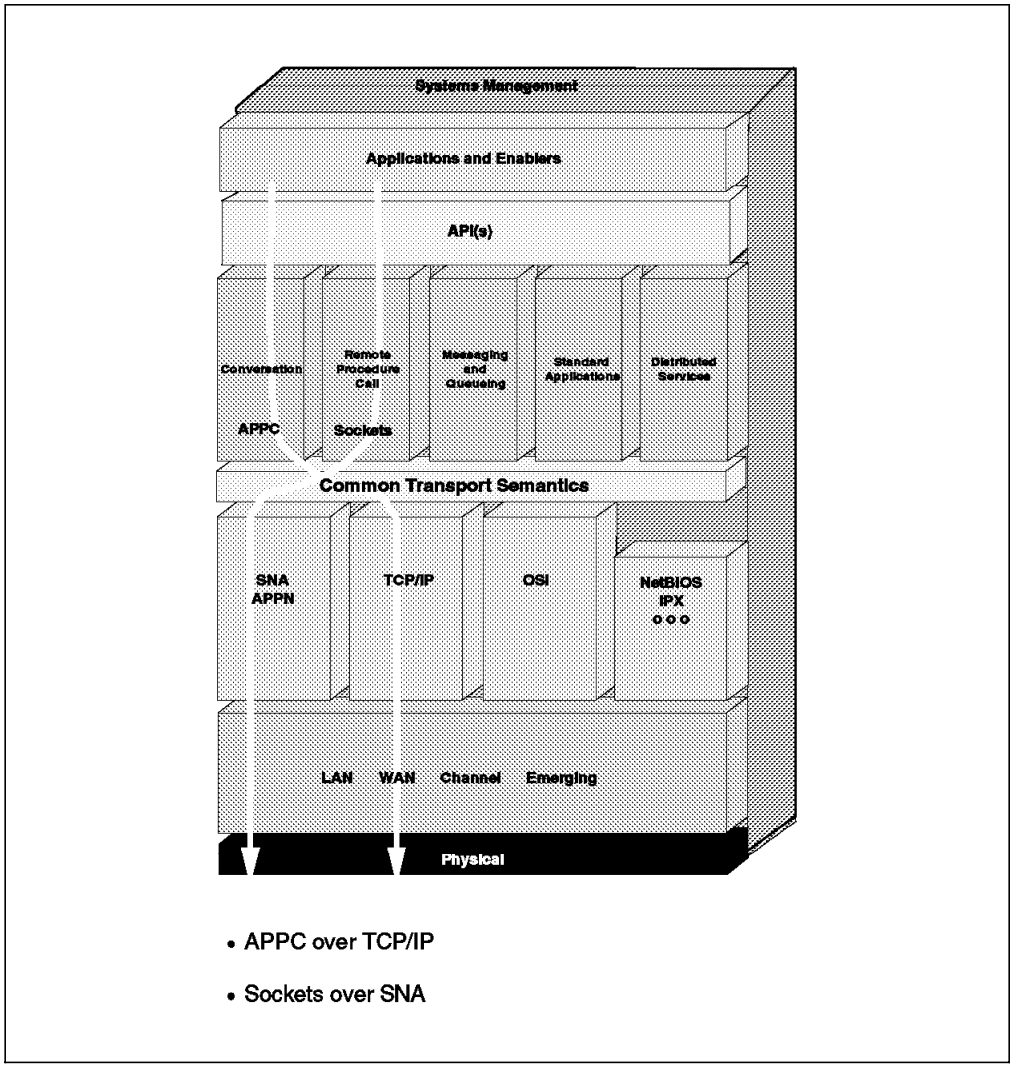


Figure 50. AnyNet/MVS - Relation to Networking Blueprint

It must be recalled that the common transport semantics concept covers the following cases:

- **Same protocol:** This is where the transport user and transport provider belong to the same protocol family, indeed there may be no separate names for the transport user layer combinations and the transport provider layer combinations. These are the native-native cases and the common transport

semantics layer is null. In other words these cases represent communication protocols as they have always been.

- **Mixed protocol:** Where the transport user layers and the transport provider layers belong to different protocol families, the common transport semantics layer is no longer null. However there are two types of common transport semantics layer solutions available:

  1. *Standards* solutions (RFCs) may already be available and have been implemented. RFC 1006 is a specification for OSI applications over TCP/IP. RFCs 1001 and 1002 are standards for running NetBIOS applications over TCP/IP; they are available in an OS/2 and DOS environment and implemented in IBM NetBIOS Version 1 for TCP/IP Version 1.2.1 for OS/2 (also available as IBM TCP/IP Version 2.0 for OS/2 - NetBIOS Kit) and IBM TCP/IP Version 2.1 for DOS - NetBIOS.
  2. *MPTN* solutions are proposed as a generalized implementation of the common transport semantics layer where standards solutions do not exist.

7.1 SNA over TCP/IP

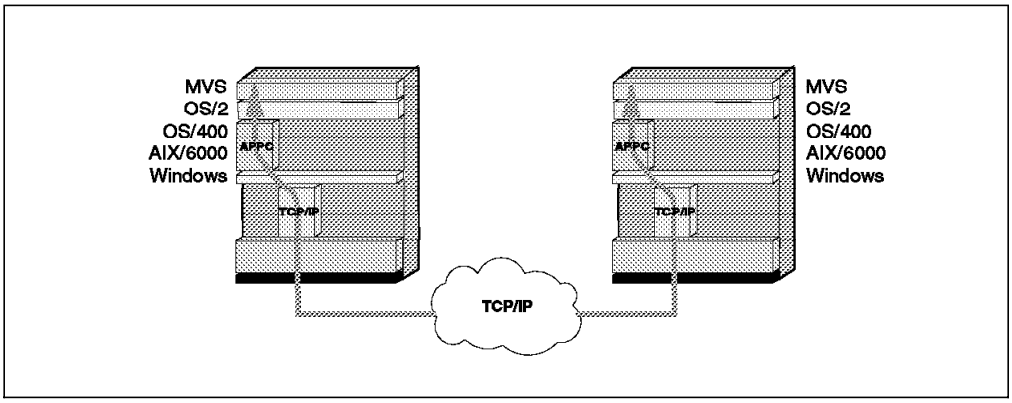


Figure 51. SNA over TCP/IP - MPTN Access Node

SNA over TCP/IP supports LU6.2 APPC communications over a TCP/IP network. A Customer Information Control System (CICS), Information Management System/Extended System Architecture (IMS/ESA), Distributed Relational Database Architecture (DRDA), DATABASE 2 (DB2), or any MVS/ESA and OS/2 APPC (LU6.2) application, is able to communicate across a TCP/IP network with an OS/2 workstation or another host that has the MPTN function installed along with the APPC API. The SNA over TCP/IP communications can be host-to-workstation, workstation-to-workstation, or host-to-host. When in a host-to-host configuration, dependent LU types 0, 1, 2 and 3 are supported, but only for PLU-initiated sessions. For dependent LU types 0, 1, 2 and 3 between a host in the SNA network and an PS/2 workstion, the dependent LU server and requester, DLUS and DLUR, functions will also be needed.

In addition to the support for the MVS and OS/2 operating system platforms, products are now available for the OS/400, AIX/6000 and Windows platforms.

Existing APPC applications will now run over the two predominant networking protocols, SNA and TCP/IP, without any changes to the application, whether a session over an IP network or an SNA network is desired. See 7.1.5, “Network

Examples" on page 103 for an overview of possible network configurations when the SNA over TCP/IP functions are used.

7.1.1 System Structure

Figure 52 shows the main components of the VTAM and Communications Manager/2 implementation of MPTN. The SNA over TCP/IP implementation in MVS/ESA supports the following program interfaces:

- VTAM's record API with LU6.2 support within the application.
- VTAM's APPCCMD interface.
- MVS/APPC and CPI-C, since both are built on top of VTAM's APPCCMD interface.
- Any subsystem using one of the preceding APIs. For example CICS, which uses the record API with the LU6.2 support within CICS, or IMS, which uses the MVS/APPC and CPI-C interfaces, are supported.

The SNA over TCP/IP implementation for OS/2 supports the CPI-C and APPC interfaces of Communications Manager/2.

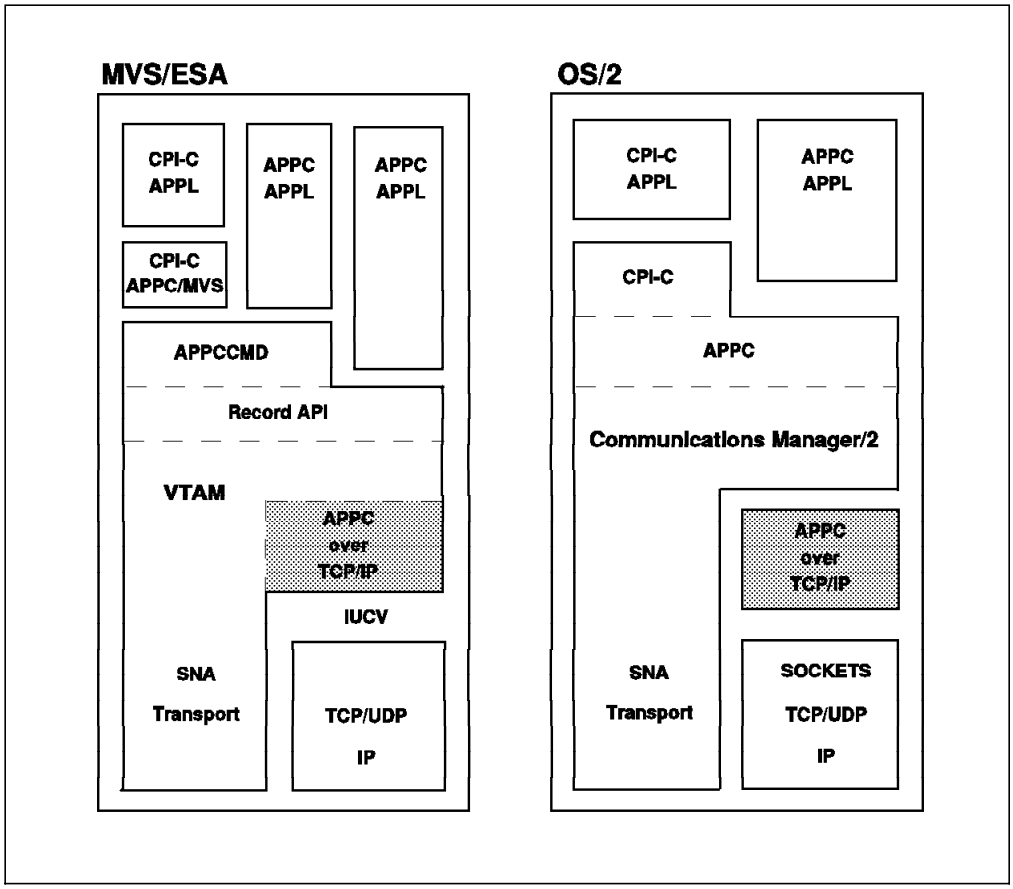


Figure 52. SNA over TCP/IP Node Structure for MVS/ESA and OS/2

SNA over TCP/IP uses the following support services provided from IBM TCP/IP Version 2 Release 2.1 for MVS and IBM TCP/IP from Version 1.2.1 for OS/2:

- IUCV socket interface in TCP/IP for MVS only
- AF_INET Socket API for C language
- TCP protocol support
- UDP protocol support
- IP protocol support
- Domain name system resolver function (DNS)

SNA over TCP/IP uses protocols that bypass the lower transport layers of the SNA architecture as seen in Figure 52 on page 94. Instead of encapsulating the entire SNA path information unit (PIU) within a TCP transmission frame, APPC builds a basic information unit (BIU) that TCP/IP can accept from SNA over TCP/IP when sending and receiving data. Consequently, the overhead of building and deciphering the SNA transport information is eliminated, and hence no transmission header, only the sequence number, the request header, and the request unit are carried over the TCP/IP connection. Data sent by an application program passes to the APPC services of VTAM or Communications Manager/2. The data passes through the SNA architectural layers and is presented to SNA over TCP/IP instead of the usual SNA transport layer. The IP hosts, which have implemented the MPTN functions in MVS VTAM or CM/2 are MPTN access nodes with a TCP/IP transport provider and an SNA transport user. The MPTN formats and protocols are used to support LU-LU sessions over the IP network.

The designated TCP/IP port (well-known local address) for exclusive use of SNA over TCP/IP enables TCP/IP to determine whether to route the transmission frame it receives to a TCP/IP application or to SNA over TCP/IP.

7.1.1.1 Differences of the Implementations in MVS/ESA and OS/2

The process of selecting the *transport provider* (SNA or TCP/IP) is different for the MVS/ESA or OS/2 implementation.

- In the VTAM implementation a *TCP major node* is used to define the IP network to VTAM. The IP network is treated as a link between Type 2.1 nodes and the TCP major node contains GROUP, LINE and PU statements, where the PU statement represents an adjacent Type 2.1 link station. The adjacent link station supports sessions to remote independent LUs through the IP network and can be attached to a node with a net ID different from the net ID of the VTAM node. Only one LINE and PU statement out of the many which can be defined may be active at any one time. This means that only one adjacent link station, which is accessed over the IP network can be active at any one time.

  A *CDRSC* (cross-domain resource) definition is required for every LU in the IP network to which sessions are to be initiated. In the CDRSC, the adjacent link station, having the name of the PU statement, through which the session is established can be defined with the ALSLIST operand. So, if the PU statement name of the TCP major node is coded on the ALSLIST parameter, the LU is reached through the IP network. There is also the possibility of using the session management exit, which then selects the route through native SNA or through the IP network. If an LU from the IP network initiates a session with VTAM, VTAM can define this LU dynamically as a CDRSC.

- In the CM/2 implementation a routing preference table is used to select the transport provider, either SNA or IP, for LUs to which an APPC application initiates a session. In this table a default is set to either native (=SNA) or

nonnative (=TCP/IP), which is then used for all LUs not explicitly defined in the table. LU names can be explicitly added, removed, or displayed.

7.1.1.2 APPC Gateway Function in AnyNet/MVS

AnyNet/MVS provides a gateway function for SNA over TCP/IP, as is illustrated in Figure 53. In fact, the VTAM implementation of the MPTN access node and the VTAM implementation of the MPTN gateway node are identical since VTAM always acts as an MPTN gateway. When performing the MPTN access node function, VTAM uses SNA flows internally from the MPTN gateway support to the origin or destination LU. If this LU happens to be supported by the same VTAM node, VTAM has the appearance of an MPTN access node.

VTAM establishes an SNA Type 2.1 link through the IP network. This means that all the possibilities and limitations of a Type 2.1 link apply to the connection between the parts of the SNA network separated by the IP network. For example, cross-domain resource manager, CDRM to cross-domain resource manager sessions cannot be established between VTAMs across the IP network. A further limitation is that, since there is no end-to-end exchange identification flow at the time the link is established, it is not possible to establish control point, CP, to control point sessions over the link. Thus the link cannot be a link within the topology of an APPN network.

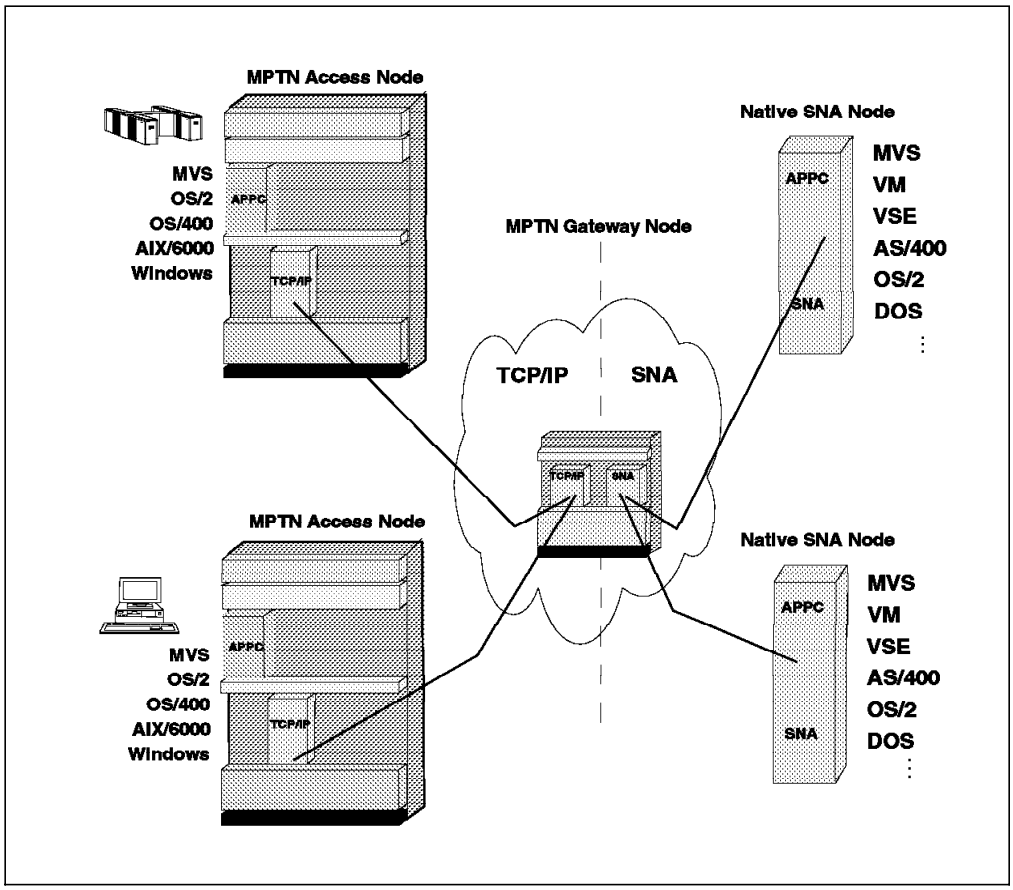


Figure 53. SNA over TCP/IP - MPTN Gateway Node

If there is an APPC session originating from the SNA side of the VTAM gateway to an APPC application located somewhere on the IP side of the VTAM gateway, the destination LU must be defined as a CDRSC in the VTAM gateway. However, if there is an APPC session originating from the IP side of the VTAM gateway to

an APPC application located somewhere on the SNA side of the VTAM gateway, the destination LU need not be defined as a CDRSC in the VTAM gateway since VTAM will create the CDRSC dynamically using information in the arriving BIND request unit.

7.1.2 Address Mapping

The native directory services used in IP are *extended* to support the MPTN addressing for SNA over TCP/IP. The IP directory services use the HOSTS.LOCAL file or the Domain Name Server (DNS) to map the host names to IP addresses (see RFC 952 for details on IP naming). For example, a HOSTS.LOCAL entry “128.100.10.2 SERVER” maps an IP host with the host name *SERVER* to an IP address *128.100.10.2*. For SNA over TCP/IP the host names are extended to include the SNA network-qualified names. If no DNS is available the HOSTS.LOCAL file in every MPTN access or gateway node which provides the SNA over TCP/IP services must be used for the address mapping mechanism. In the following discussion we use the implementation of the Domain Name Server in IP for the description of the the Domain Name Server in IP for the description of the address mapping mechanism.

The net ID used in SNA as part of the transport-user address corresponds one to one to a subdomain in the IP network used as part of the transport-provider address.

7.1.2.1 Registering an SNA Name

The registration process for SNA names (transport user address in the MPTN architecture) is the same as for IP domain names, because the native IP directory services are used. This is a manual process rather than the programmable registration process which MPTN architecture ideally envisages.

Each *destination* LU name must be mapped to a corresponding IP address in the DNS. Consequently each LU name must have a corresponding resource record (RR) defined in a DNS resource set. The LU name need be defined only in one DNS which can be reached by all related MPTN access nodes. The LU names are defined in the DNS like any other IP hosts as an IP domain name. For example, there is an APPC application associated with the LU name LU1APPC with the SNA net ID NETA, and there is already a subdomain in the IP network with the name SNA.IBM.COM. Then a new subdomain for NETA.SNA.IBM.COM must be defined and within this subdomain the LU name is defined, like any host within a subdomain, with a domain name *LU1APPC.NETA.SNA.IBM.COM* and with the corresponding IP address *128.100.10.3*.

7.1.2.2 Resolving SNA Names

When VTAM or CM/2 initiates a session for an application program, it translates the SNA address (network-qualified name) into IP routing information (IP address) and uses the IP address to create a TCP connection to the appropriate system. The MPTN functions in VTAM and CM/2 use the TCP/IP transport provider for name resolution. A *gethostbyname* is issued to the TCP/IP resolver, which queries the DNS to obtain the IP address for the domain name. This name is created using the SNA LU name, the SNA net ID and the domain name suffix. This suffix is defined during configuration of VTAM (TCP major node), or CM/2 (in CONFIG.SYS). The well-known TCP/IP port number is by default 397. If this is changed it must be changed in all MPTN access or gateway nodes connected to this IP network.

The TCP/IP transport provider knows the IP address of the nearest DNS from definitions provided during the configuration. To reach the DNS, UDP/IP datagram communication or a TCP/IP connection can be used to resolve the LU name into an IP address.

7.1.3 Dependent LU Support

SNA over TCP/IP for MVS/ESA does *not* offer *direct* full support for dependent LUs. Full support for dependent LUs requires the ability for the secondary LU in the session to initiate the session. However, if the dependent LU requester and server, DLUR and DLUS, support in CM/2 and VTAM are used in conjunction with the AnyNet/2 MPTN access node product and the AnyNet/MVS product, behaving as an MPTN access node or an MPTN gateway node offering access primary LUs in the attached SNA network, support for sessions initiated by secondary LUs is provided.

The significance of using this combination of functions is that there is no longer a need for sessions between the VTAM SSCP and the dependent LUs to flow over the Type 2.1 link superimposed on the IP network. These sessions, needed for session initiation by the secondary LU, are encapsulated in APPC sessions connecting the DLUS and DLUR functions in their respective nodes and flow naturally over the Type 2.1 link. All necessary preliminary flows occur over the DLUR-DLUS sessions and the control point functions, SSCP or CP, of the VTAMs owning the PLU and coresiding with the DLUS function. Finally, the primary LU in the session will initiate the session itself with a BIND.

Although the DLUR and DLUS functions are normally associated with APPN configurations, it is possible to establish sessions between nodes containing the DLUR and DLUS functions over a Type 2.1 link for which there is no APPN control point connectivity.

For information on how to define DLUR support in VTAM, see *VTAM Network Implementation Guide* and *Guide to SNA over TCP/IP* manuals for VTAM Version 4 Release 2. In addition, see *AnyNet: SNA over TCP/IP Installation and Operation*, GG24-4395.

7.1.4 Command and Data Flows

The following figures show the relation of SNA over APPC command and data flows to the MPTN flows over the IP network.

In Figure 54 on page 99:

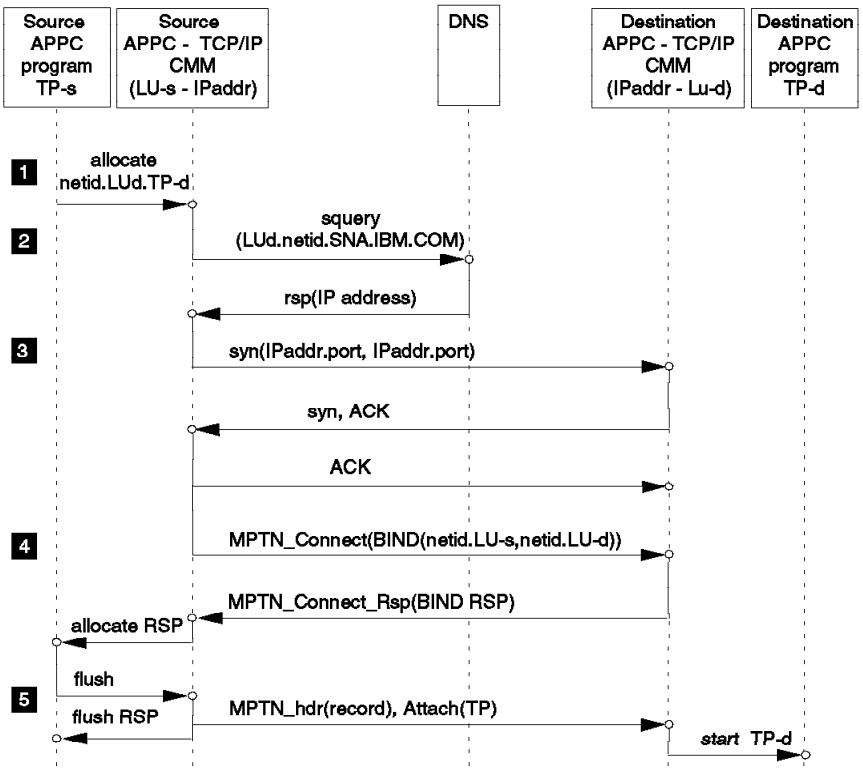
1 The APPC-s program allocates a conversation with the destination program Netid.LU-d.APPC-d. There is no session established between the two LUs (Netid.LUs, Netid.LUd).

2 The SNA services must first establish a session which is routed over the IP network. The MPTN functions of SNA over TCP/IP first map the destination SNA LU-d name to an IP address. The domain name server returns back the IP address.

3 Next the TCP/IP connection which will eventually carry the MPTN connection is established to this destination IP address using the well-known port number of SNA over TCP/IP.

4 Over this connection an MPTN_Connect is sent, which carries the SNA BIND as data, and a positive response (including the BIND response) is sent back if destination LU-d accepted the BIND. The allocation response then is returned to source APPC-s program.

5 The program must flush its send buffer and request to send the SNA attach (FMH-5), which is sent by MPTN with an MPTN header over the IP network to the destination. MPTN functions at the destination then transfer the attach to SNA LUd, which starts the destination APPC-d program.



Legend:
DNS: Domain Name Server
SYN: TCP packet with SYN bit set
ACK: TCP packet with ACK bit set
SYN,ACK: TCP packet with SYN and ACK bits set

Figure 54. APPC Connection Setup over TCP/IP

In Figure 55 on page 100:

1 the APPC-s program allocates a conversation with an APPC-d program, which is located in the IP network. The APPC-s program is not in the same SNA host (SNA domain) as the APPC gateway to the IP network.

2 Normal subarea SNA commands (INIT-SELF, CDINIT, CDCINIT, and CINIT) flow between the source LU, the source SNA host, and the gateway. The VTAM APPC gateway knows the location of the destination LU-d from its CDRSC definition.

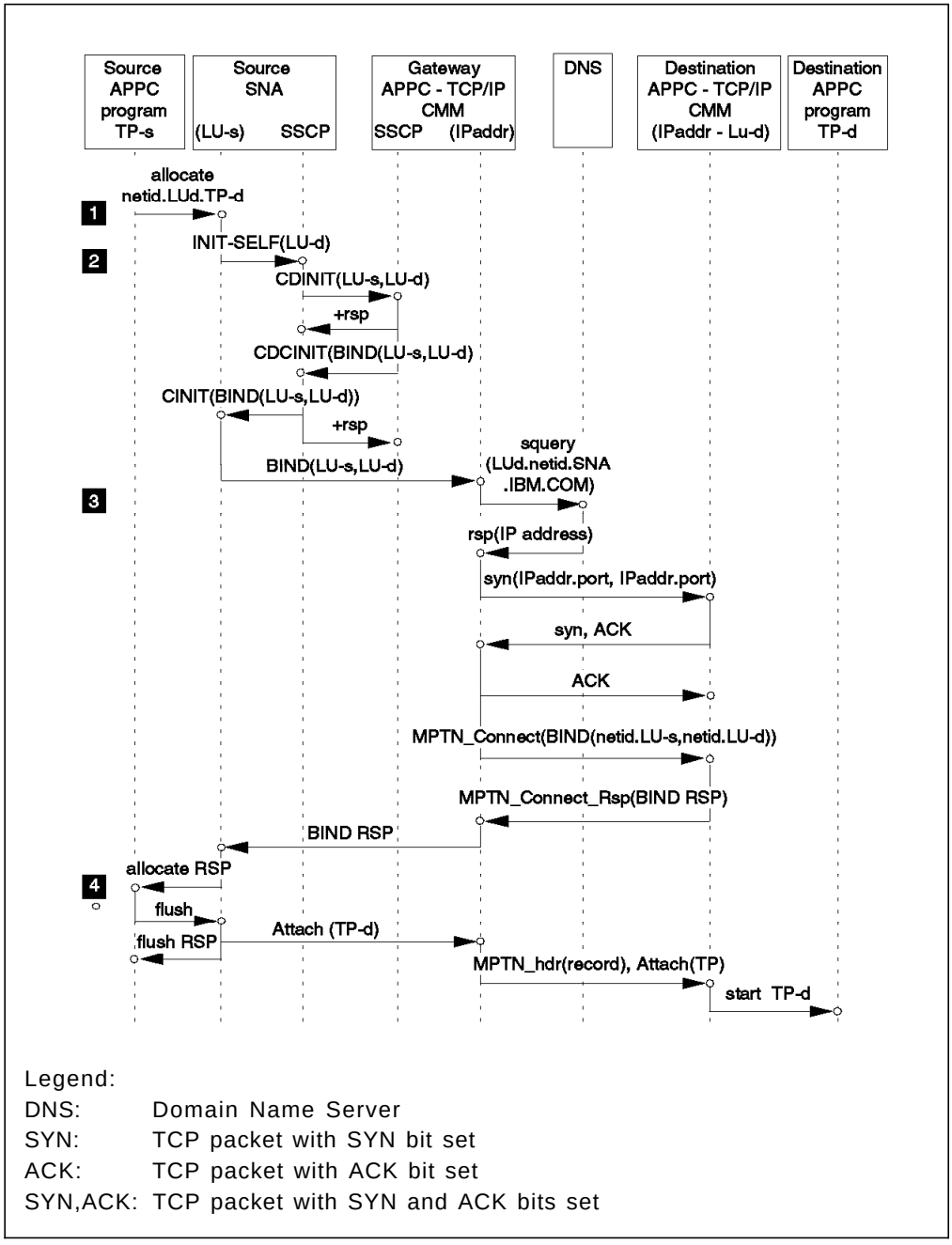


Figure 55. APPC Connection Setup over TCP/IP with APPC Gateway (Subarea Flows)

3 It initiates the session to the destination LU-d over the IP network as in the previous figure, when the BIND for the destination LU-d is received at the gateway.

4 After the BIND response is sent back to source LU-s, the APPC-s program receives the allocate response. The program flushes its send buffer and in this way initiates the SNA attach for the destination APPC-d program. The gateway sends the attach in the with an MPTN header over the TCP connection across the IP network to the destination node.

Figure 56 gives an overview of MPTN functions involved when SNA data is sent over an IP network. To maintain the record boundaries an MPTN header is required for all segments of a record which are sent over this conversation.

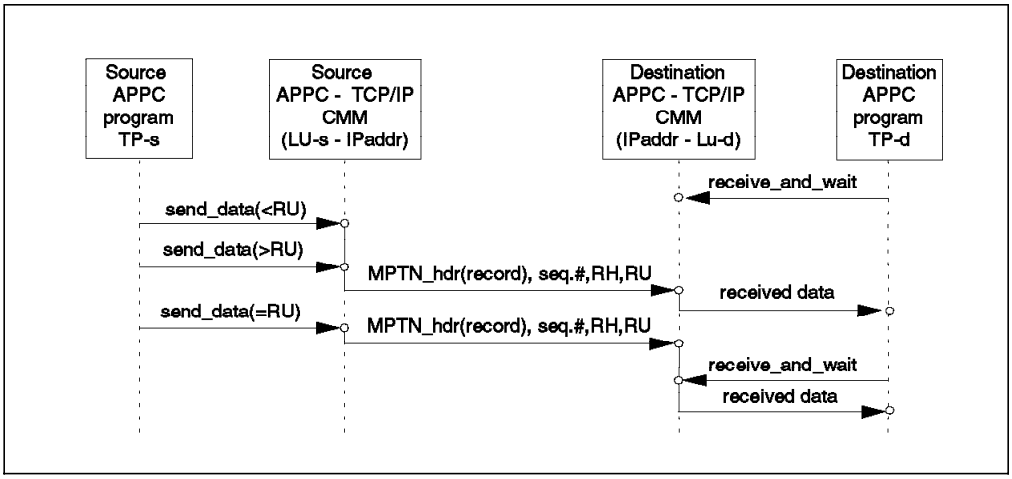


Figure 56. APPC Data over TCP/IP

Figure 57 gives an overview of MPTN functions involved when a conversation is closed between two APPC programs in an SNA over TCP/IP configuration. Only a termination request is sent by the source APPC-s program which is included in the MPTN format as a Conditional End Bracket (CEB) and sent over the TCP/IP session. The LU 6.2 session is not terminated.

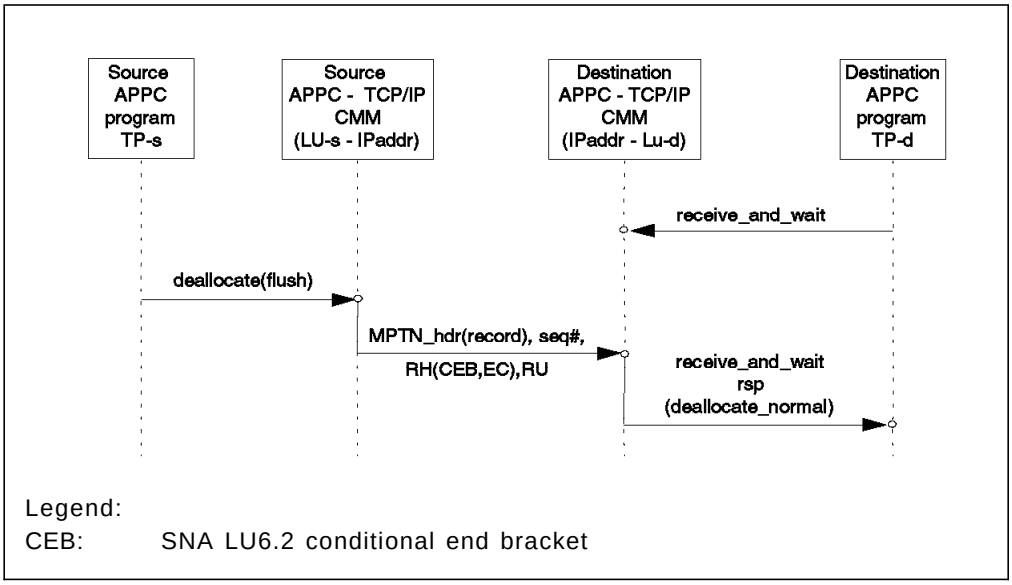
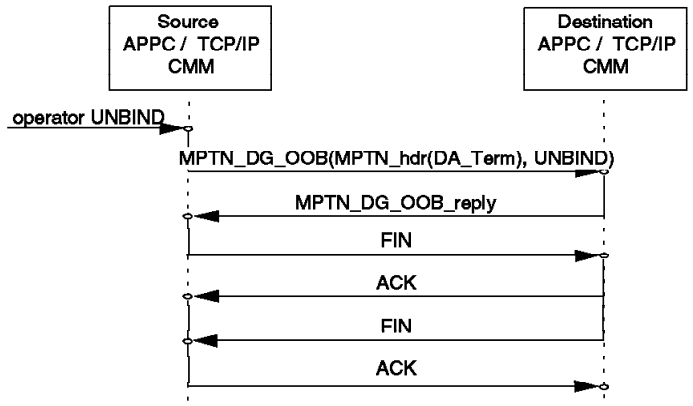


Figure 57. APPC Deallocate a Conversation over TCP/IP

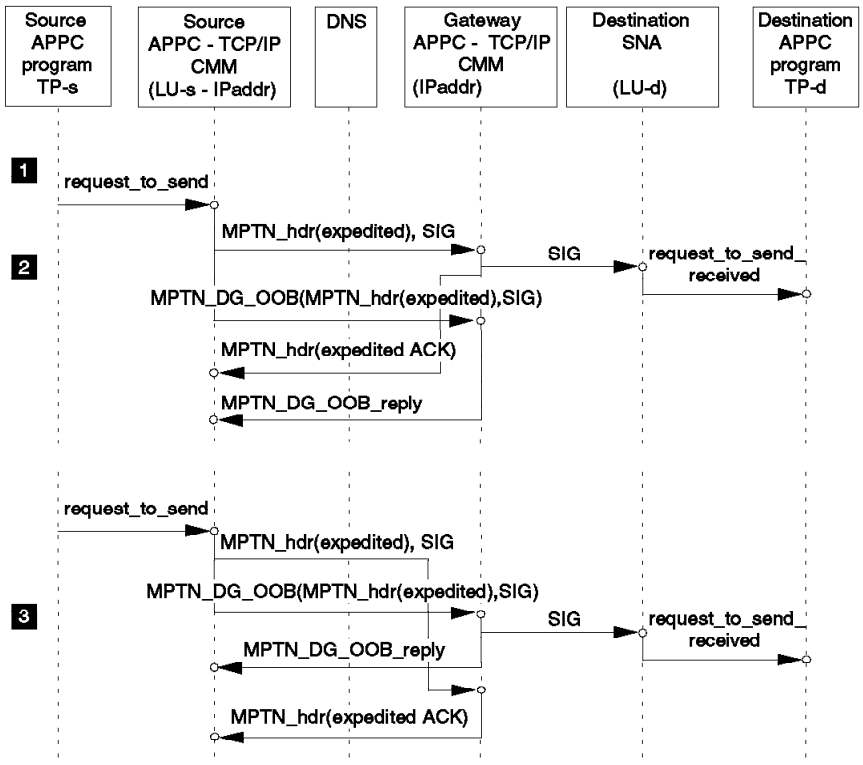
In Figure 58 on page 102 the termination of a session between LUs is shown. The application programs using the session for conversations do not terminate a session. Sessions may be terminated by operator request. When the UNBIND request flows, first the SNA session over the IP network is terminated, then the MPTN and TCP connection between the MPTN nodes is terminated.



Legend:
FIN: TCP packet with FIN bit set
ACK: TCP packet with ACK bit set

Figure 58. Operator UNBIND for SNA over TCP/IP

Figure 59 shows the flow of expedited data traffic of an APPC conversation over an LU 6.2 session across a TCP/IP network. The APPC gateway function of VTAM is used to connect both networks.



Legend:
SIG: SNA SIGNAL command (request to send for LU6.2)

Figure 59. Expedited Data Traffic in SNA over TCP/IP

1 The APPC-s program requests a send direction from the remote APPC-d program. This request is sent in a SIG frame in the SNA network

2 The SIG is sent over the IP network with an MPTN header via the TCP/IP connection and, if no acknowledgment is received in time, via a UDP/IP datagram. The connection data first arrives at the remote APPC gateway and MPTN transforms this data to normal SIG SNA format in the SNA network.

3 If the datagram arrives first, the datagram information is forwarded to the SNA network.

Note that only one SIG is forwarded to the destination.

7.1.5 Network Examples

The following examples show possible configurations using AnyNet/MVS and AnyNet/2.

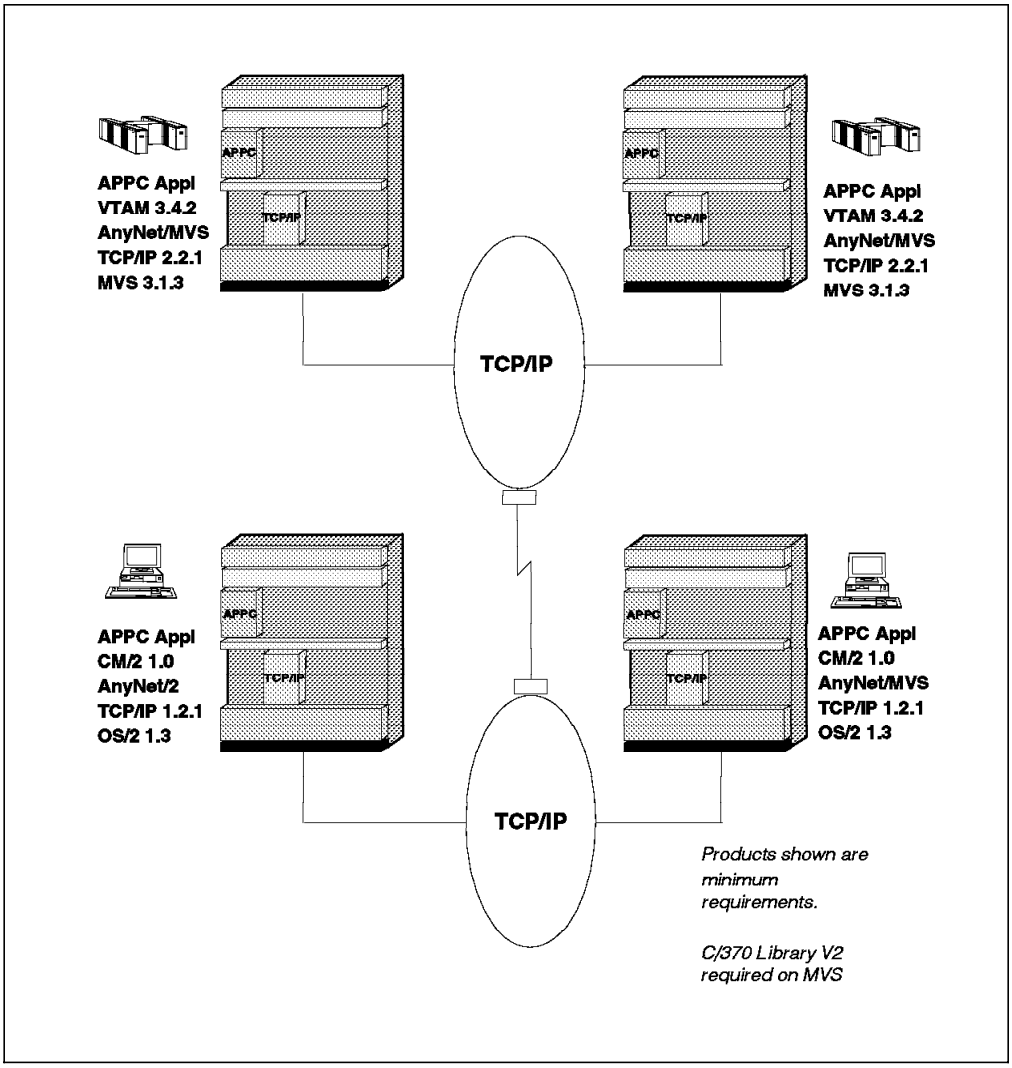


Figure 60. A Single IP Network with SNA over TCP/IP

In Figure 60 the APPC applications in OS/2 and VTAM communicate with each other using the normal APPC protocols. The AnyNet products transfer the SNA protocols over the IP network using MPTN protocol. Independent LUs are

supported without restriction. Dependent LUs are supported between host nodes with the restriction that the session must be initiated by the primary LU. Dependent LUs are supported between host and OS/2 nodes, including the possibility for the secondary LU to initiate the session. This requires the dependent LU server (DLUS) function in the host and the dependent LU requester (DLUR) function in the OS/2 node. The primary LU (host application) may be in another host node throughout the SNA network. Dependent LUs (LU 0, 1, 2, 3) in the VTAMs can be reached over the IP network only if the primary LU initiates the session.

The native address of the LUs in the IP network is the IP address of the node containing the LU. The native directory services of TCP/IP are used to map the LU names to the IP addresses. Routing in the IP network is based purely upon the IP routing mechanism, based on tables either static or dynamic when supported by routing protocols.

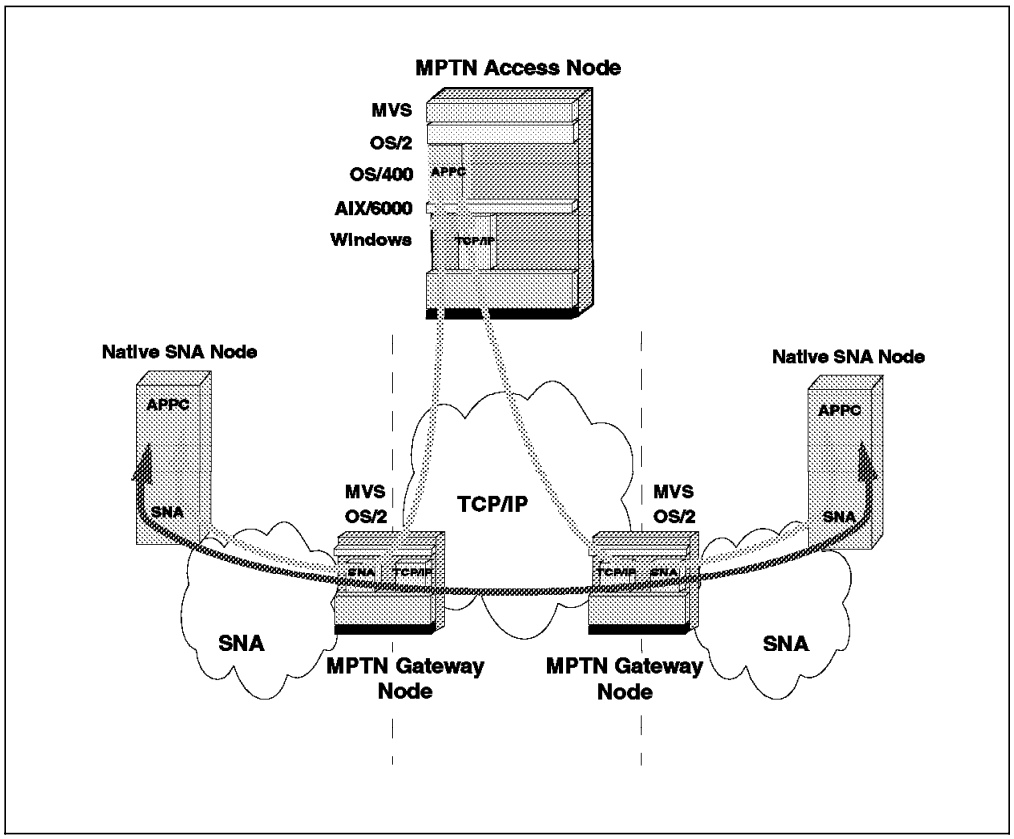


Figure 61. IP and SNA Network Connected through an SNA over TCP/IP MPTN Gateway

Figure 61 reflects a possible configuration utilizing the SNA over TCP/IP gateway function. An APPC application running in an MPTN access node communicates with matching APPC applications in native SNA nodes through the APPC gateway. The gateway provides the TCP/IP to SNA conversions so that it is not necessary to have MPTN functions installed in every SNA node.

The sample network also shows that two SNA over TCP/IP gateways can be used to provide connectivity and session transport to APPC applications in native SNA nodes through a (backbone) IP network. The SNA application need not implement any MPTN or TCP/IP functions; they are not even aware that an intermediate IP network is used to transport their sessions' traffic.

In the SNA network, SNA routing is used, in the IP network, IP routing is used. If more than one SNA over TCP/IP gateway exists:

- SNA protocols are used to route to the SNA over TCP/IP gateway in the SNA network.
- IP protocols are used in the TCP/IP network to route to the SNA over TCP/IP gateway. The LU name to IP address mapping is defined at configuration time, so the selected SNA over TCP/IP gateway for the destination LU is defined at this time.

7.2 Sockets over SNA

Sockets over SNA offers a sockets application programming interface (API), specifically the internet address, AF_INET, family sockets API, as the transport user interface and uses APPC for the underlying transport provider function. In the case of the MVS AnyNet product, the complete sockets environment is created independently of the TCP/IP product for the platform. In the case of the other sockets over SNA products, the TCP/IP product associated with the platform provides the sockets environment. This is important for the protocol and names files required to support calls such as *getprotobyname*. AF_INET sockets typically provide an interface to a TCP/IP network. Sockets over SNA AnyNet products will provide organizations that currently use SNA networks, but who also have access to the Internet, access to the large body of applications which use TCP and UDP over the Internet for relatively small initial and long-term costs. The majority TCP and UDP applications that use internet addressing and *do not use the broadcasting* should be usable in the sockets over SNA environment.

Sockets over SNA does not require any changes to existing AF_INET sockets applications; hence no programmer time is needed to modify socket applications to run on an SNA network. Sockets over SNA provides the same connectivity as does the SNA implementation on each of the platforms.

The sockets over SNA function delivered with AnyNet/MVS consists of host VTAM code and OS/2 code which is downloaded to the workstation. The OS/2 code can also be ordered and installed separately with AnyNet/2.

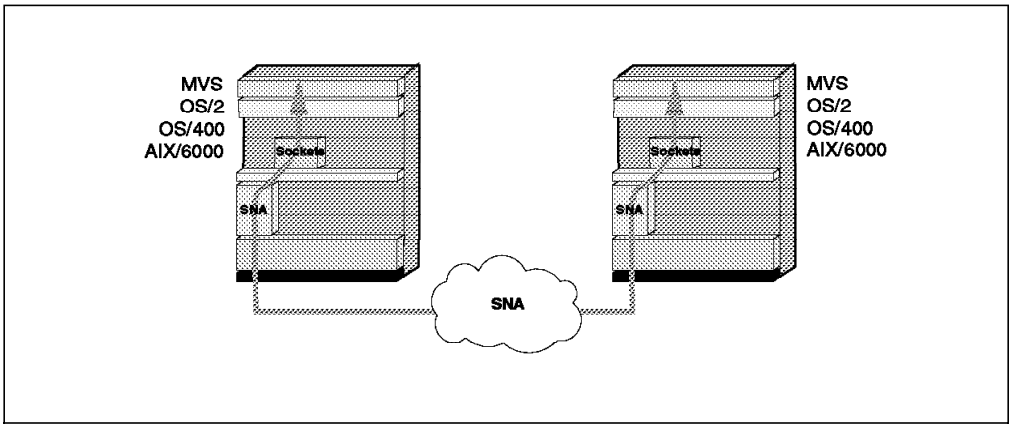


Figure 62. Sockets over SNA - MPTN Access Node

Figure 62 illustrates the sockets over SNA MPTN function. The MPTN user can now select desirable socket applications for use in an SNA network without concern for the details of the network transport layer. The sockets over SNA

AnyNet products contain the network transport conversions and compensations necessary to enable this.

7.2.1 Application Program Support (MVS/ESA)

Sockets over SNA supports application programs written to the AF_INET C sockets interface for IBM TCP/IP for MVS. On MVS/ESA, existing applications have to be relinked to use sockets over SNA. The following applications have been tested for compatibility with sockets over SNA on MVS/ESA:

- Network File System (NFS)
- X Window System (X-Windows) client support environment

The following are provided to enable application programs to use sockets over SNA on MVS/ESA.

- Description of the sockets interface supported by sockets over SNA on MVS/ESA. This description also outlines the differences between the sockets over SNA interface and the IBM TCP/IP for MVS interface.
- Header files needed to compile a sockets application program.
- Object code that is linked with a sockets application.
- Required run-time module.

7.2.2 Application Program Support (OS/2)

Sockets over SNA supports application programs written to the AF_INET interface for IBM TCP/IP for OS/2 (for example, TELNET and FTP). These applications do not have to be recompiled or relinked. The following IBM TCP/IP applications have been tested for compatibility with sockets over SNA on OS/2:

- File transfer protocol (FTP and FTPPM)
- Virtual terminal protocol (TELNET)
- TALK
- Remote execution protocol (REXEC)
- Remote shell execution (RSH)
- Packet internet groper (PING)
- HOST
- X window system (X-Windows)

7.2.3 Sockets over SNA System Structure

The system structure of the MVS/ESA and OS/2 workstation nodes are shown in Figure 63 on page 107. In the MVS/ESA node, the sockets over SNA function runs in its own address space and is seen by VTAM as a normal APPC application. In the OS/2 workstation the sockets over SNA function runs as an application using CM/2′s APPC interface.

Sockets over SNA presents a C socket interface to C application programs. When an application program attempts to open a stream connection with, or send a datagram to, another application program using sockets over SNA, it supplies the same information that it would supply to the C socket interface of the TCP/IP product.

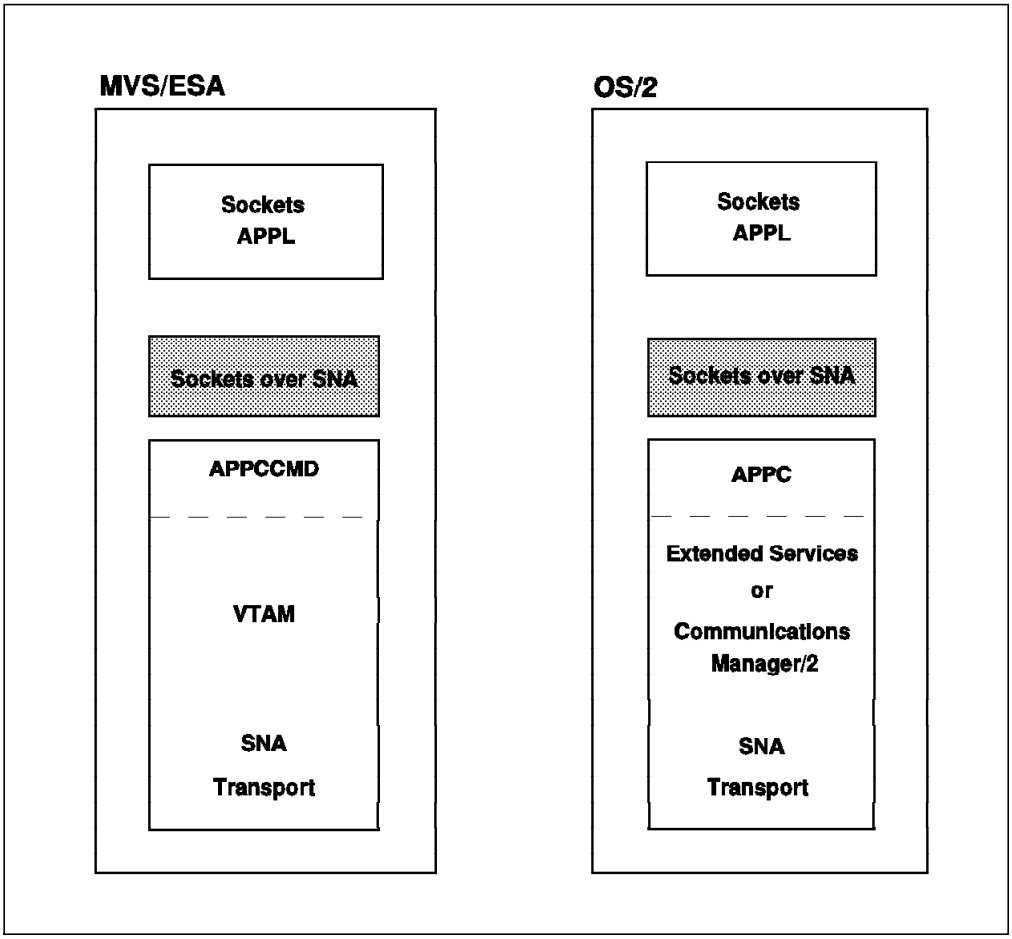


Figure 63. Sockets over SNA - System Structure

Operating as an LU 6.2 application, sockets over SNA receives sockets calls through its application programming interface and generates LU 6.2 calls that correspond to the sockets calls. Sockets over SNA issues LU 6.2 calls to its APPC interface.

On MVS/ESA, sockets over SNA issues APPCCMD calls to VTAM. On OS/2, sockets over SNA issues APPC calls to Communications Manager.

7.2.4 Mapping Sockets Calls to LU 6.2 Calls

When an application program invokes sockets over SNA to establish a stream connection with another application program, sockets over SNA establishes two half-duplex LU 6.2 conversations for each stream connection it sets up. Each LU 6.2 conversation services one direction of the duplex connection. The conversations are deallocated when the stream socket connection is ended. If the the underlying APPC implementation supports full-duplex APPC, only one full-duplex LU 6.2 conversation will be allocated.

One LU 6.2 conversation is established for all datagrams sent to a single destination unless such an LU 6.2 conversation has been retained following the sending of the previous datagram. Conversations dedicated to datagram traffic are deallocated if they are unused for some a period of time. which may be a parameter of the sockets over SNA AnyNet product customization.

7.2.5 Sockets over SNA and TCP/IP Coexistence

For both OS/2 and MVS/ESA, sockets over SNA and TCP/IP can be installed and operated on the same node. However, sockets over SNA and TCP/IP interoperate differently in the cases of an MVS/ESA node and on an OS/2 node.

On MVS/ESA:

- The interface addresses used by sockets over SNA and TCP/IP are *not* associated with the same IP implementation.
- Sockets over SNA and TCP/IP operate independently of one another and have no awareness of the other's presence on the node.
- When the sockets application program is linked, it is linked to either sockets over SNA or to TCP/IP and, therefore, can only use one or the other.

On OS/2:

- The interface addresses used by sockets over SNA and TCP/IP are associated with the same IP implementation.
- Sockets over SNA and TCP/IP are integrated. When sockets over SNA is installed, a common device driver services all sockets application programs.
- A sockets application program can communicate with a sockets over SNA application program and a TCP/IP application program concurrently.

7.2.6 AnyNet/2 Sockets over SNA Gateway

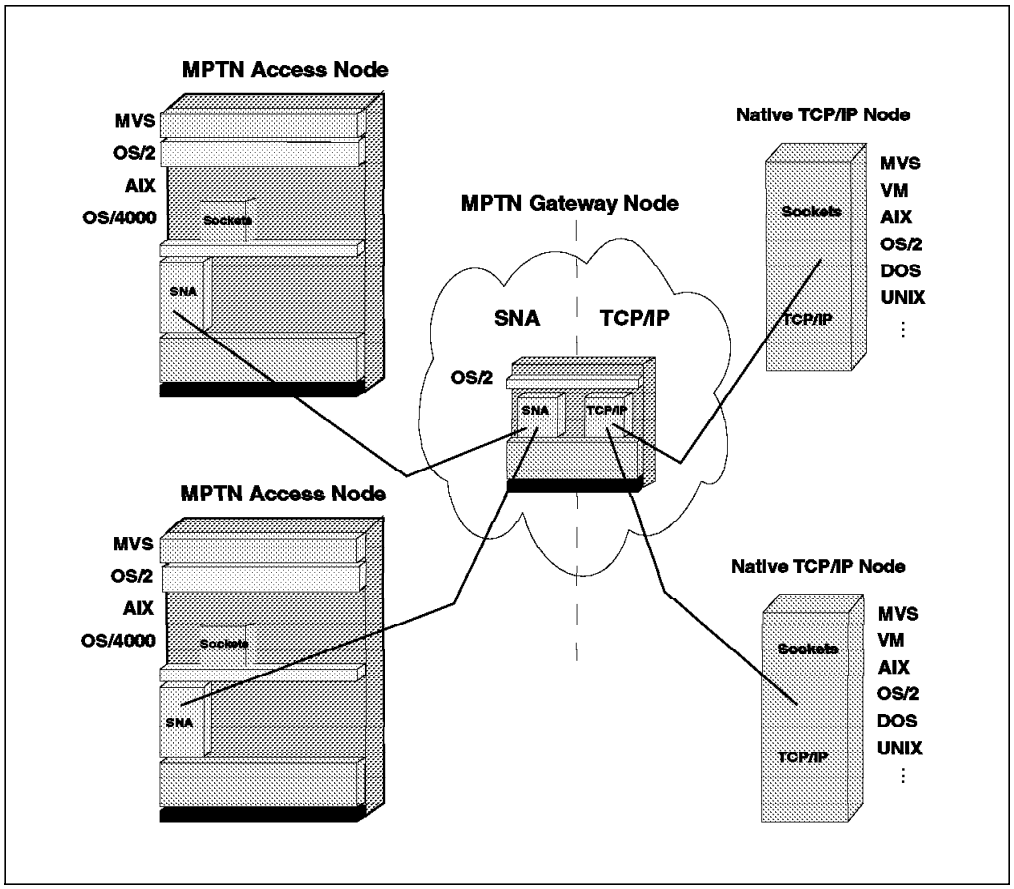


Figure 64. Sockets over SNA - MPTN Gateway Node

The AnyNet/2 Sockets over SNA Gateway is shown in Figure 64. In a manner very similar to the SNA over TCP/IP gateway, this sockets gateway allows applications running in MPTN access nodes to communicate with matching sockets applications running in TCP/IP nodes which do not have AnyNet functions installed. When using two gateways, also two sockets applications running in TCP/IP nodes which do not have AnyNet functions installed can communicate across a backbone SNA network. Such TCP/IP nodes can be any hosts supporting sockets applications over TCP/IP.

7.2.7 Address Mapping

A node implementing sockets over SNA functions is an MPTN access node and is identified by both an internet address and an SNA address. An application program uses internet addresses to tell sockets over SNA what addresses are associated with its connections, just as if it was interfacing to TCP/IP. Sockets over SNA uses SNA LU names to specify source and destination locations because it operates as an LU 6.2 application using SNA transport. Because an application has no knowledge of SNA addressing, sockets over SNA has to map all incoming and outgoing internet addresses to SNA LU names.

Some of the requirements that need to be completed in this regard are:

- To define an internet address for for the sockets over SNA interface in each sockets over SNA access or gateway node network
- To determine what SNA net ID is in use in the SNA network over which sockets over SNA will operate network
- To ensure every node in the sockets over SNA network is represented by an SNA LU name which identifies the sockets over SNA function
- To ensure every internet address in the sockets over SNA network is mapped to the SNA LU name

7.2.7.1 Mapping Internet Addresses to LU Names

There are two methods which can be used to map internet addresses to LU names:

- Define an LU name for each node in the network individually. It is recommended you use this method only if a small number of nodes are using sockets over SNA or you are setting up the network for the first time and want to test communications among a few nodes.

  On every node, when sockets over SNA is started, the internet address and LU name for each possible destination node must be defined. For example, in a sockets over SNA network of 20 nodes, each node would have to define the IP address and LU name for each of the other nodes with which it wants to communicate (potentially 20 x 19 = 380 definitions in all nodes together for this example). In addition, each time a node is added to the network, all nodes that want to communicate with the new node would have to define the IP address and LU name for the new node when they start up.

- Allow sockets over SNA to generate LU names algorithmically. This method is recommended to set up LU names for networks made up of more than a few nodes. Using algorithmically-generated LU names greatly simplifies the address mapping process.

  First, a naming convention for the LU names representing sockets over SNA hosts in the network must be decided. For example, assume all LU names used by sockets over SNA are to begin with the characters “SX.” We would

define “SX……” as the LU name template on each node when the nodes are initialized to use sockets over SNA. Using the LU name template, each node has to include only one LU entry, which then covers the naming requirement for itself as well as all other nodes using the sockets over SNA network. Using the “SX” name prefix, sockets over SNA generates the remaining six characters of the LU name using the host ID portion of the internet address. When a node is added to the network, it need only complete its own local configuration to become an operational part of the network.

7.2.7.2 Comparing Internet Addressing to SNA Addressing

Sockets over SNA supports sockets applications that use internet addressing. When sockets over SNA receives an internet address, it translates (maps) the address into an SNA address that can be carried over SNA transport.

An internet address is a 32-bit number that consists of two fields: a *net ID*, although this is very often replaced by a *subnet ID* which incorporates the net ID, and a *host ID*. The net ID or subnet ID field identifies a physical network and the host ID field identifies a particular host attached to that network.

An SNA address consists of a source and destination network name, a source and destination LU name, and a transaction program (TP) name. The network name and LU name are used together to identify an access point in an SNA network. Both the network name and LU name are one to eight characters long. Sockets over SNA maps an internet address to an SNA net ID and LU name and always uses the same transaction program name (TPN). The TPN name (well-known address) is hard-coded to X¢28F0F0F1¢.

7.2.7.3 Address Mapping Process

Sockets over SNA maintains an *internet-LU mapping table* which specifies how to map IP addresses to SNA LU names and the reverse. As described above, all the mappings may be defined individually or algorithmic mapping may be used.

The following is the process that maps an internet address to an SNA name for a destination host when either technique is used:

1. A sockets application passes the internet address of the destination host to sockets over SNA.
2. Sockets over SNA queries the internet-LU mapping table to determine:
   - Which portion of the internet address to use as the IP net ID or subnet ID
   - Which SNA net ID corresponds to the IP net ID or subnet ID portion of the internet address
   - Which LU-name template is used in calculating the LU name suffix
   - Which portion of the internet address to use as the IP host ID and, hence, to calculate the remainder of the LU name

   Note: When the LU names have been defined individually, the LU name suffix is, in fact, the complete LU name and there is no calculation of the remainder of the LU name. This illustrates that, although it is intended that, when algorithmic mapping is used, the mask used to define the net ID or subnet ID and, hence, the host ID should be the same as is used for IP routing, this is not required; there is, in fact, no necessary connection between the two uses of the subnet mask.

3. Once the LU name has been determined from the information in the internet-LU mapping table, the SNA net ID, the LU name, the TP name and the SNA mode name are used by sockets over SNA to make the connection or send the datagram.

7.2.7.4 Routing an Internet Address through the Sockets over SNA Interface

When sockets over SNA is started, the network interface for the sockets over SNA function (*sna0*). through which data is routed is defined. Assigning an internet address to the sna0 interface activates the interface of the local host; sockets over SNA starts monitoring sna0 for information destined for the specified address.

The interface name sna0 is used in the same way as other interfaces within the IP routing implementation. Thus when the usual IP routing mechanisms identify sna0 as the interface over which to route an IP packet, sockets over SNA will handle the packet, determine the LU name of the destination node running sockets over SNA and finally route the IP packet over SNA using MPTN architecture.

This operates within MVS in the same way as, say, OS/2. However in the case of OS/2, there will be alternative interfaces available for IP routing since sockets over SNA and the TCP/IP product work together. In the case of MVS, if the routing does not match the sna0 interface or the loopback interface, *lo* it is an error.

7.2.7.5 Querying the DNS for Host Name-to-Address Translation

Sockets over SNA supports host name translation on MVS/ESA and OS/2. On MVS/ESA and OS/2, the name-query function, that is, the client logic which is capable of accessing the DNS, is supplied as part of sockets over SNA. Sockets over SNA does not include support for the DNS itself. This is to be expected since implementation of a DNS is often assigned to a dedicated UNIX machine such as an RS/6000.

If IBM TCP/IP is installed on OS/2, the DNS function supplied by TCP/IP can be used locally. However, the DNS function supplied by TCP/IP on MVS/ESA cannot be used. Any sockets over SNA product can use the DNS function on another node supporting sockets over SNA, except MVS sockets over SNA, when the address of the sna0 interface is used to access the DNS.

Any sockets over SNA product, except MVS, has the full use of the TCP/IP product and can thus access the DNS function on a remote node supporting the DNS function over an IP network.

If no DNS is specified to a sockets over SNA application program, or if an application cannot access a DNS to resolve a host name to an internet address, the application program attempts to query a local HOSTS file.

7.2.8 Adding Value - Selection of SNA Mode Table According to TCP/UDP Port

One of the facilities offered by sockets over SNA implementations is to exploit the benefits of SNA flow control and its ability to favor one session over another in routing in intermediate nodes.

In SNA, the characteristics of a session which affect, most importantly, performance are grouped together and assigned a *mode name*. Defining these

characteristics is a customer choice and it allows sessions to be assigned performance parameters which match the intended business use of the function represented by the session associated with a particular mode.

Sessions for all LU types are characterized by the mode name with which the session is bound. In the case of LU 6.2 sessions, conversations are also associated with mode names and a conversation will be assigned only to a session bound with the corresponding mode name.

Although IP headers define a *type of service* field, this is very rarely used and is not appropriate as a means of mapping performance characteristics to SNA sessions. On the other hand, TCP/IP defines well-known ports for specific TCP/IP applications such as FTP, file transfer, and TELNET, interactive command entry and response.

Sockets over SNA implementations provide for a mapping from the well-known port to the mode name to be used for conversations supporting that application over the SNA network.

A typical assignment would be as described in Table 7.

Table 7. Typical Assignment of Ports to Mode Names

port	**function**	**mode name**
20	FTP data transfer	#BATCH
21	FTP interactive	#INTER
23	TELNET	#INTER

7.2.9 Command and Data Flows

The following figures show some command and data flows for the Sockets over SNA implementation. AnyNet products use APPC, so the SNA flows are shown as APPC verbs.

In Figure 65 on page 113 a TCP/IP connection is established over the SNA network.

1 The sockets application program SO-s connects to the destination sockets application SO-d after this program has issued a *listen*.

2 Sockets over SNA translates the destination IP address to the LU-d name. Then it issues an *allocate* to the destination LU-d and the destination TP name of the Sockets over SNA implementation (CMM-d). If no SNA session to LU-d existed before, a session is established and a conversation is initialized. Otherwise, only a new conversation is initiated to CMM-d. CMM-d gets ready to receive data over the conversation and the *MPTN_Connect* request for SO-d is sent from CMM-s.

3 To have a full-duplex socket connection over the SNA session between LU-s and LU-d, CMM-d allocates a conversion back to CMM-s and returns the MPTN response for the *connect* request. In CMM-d the *connect* from SO-s is queued for SO-d.

If the APPC implementations had both been capable of supporting full-duplex APPC, only one Allocate would be necessary to enable the full-duplex socket connection to be handled by APPC flows.

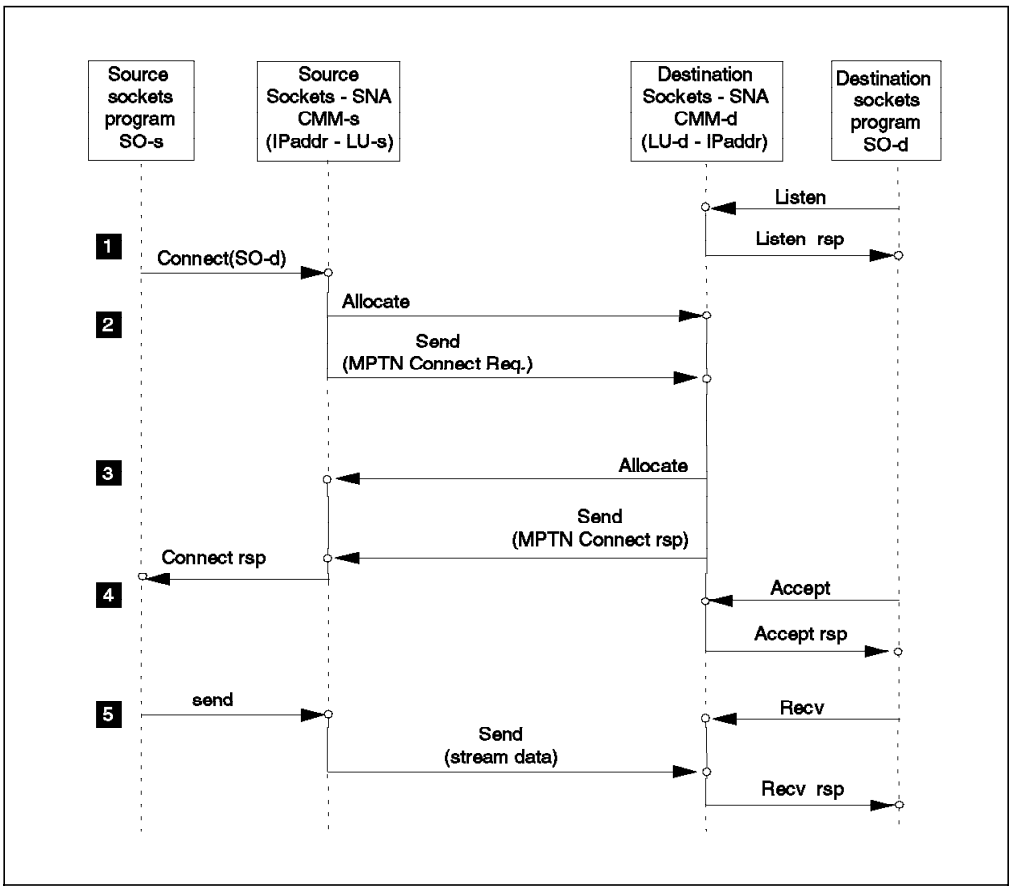


Figure 65. Connection Establishment and Sending Data in Sockets over SNA

4 CMM-s returns the response for the *connect* to SO-s. The destination SO-d is informed of the queued connection and issues an *accept* (which completes immediately) and the connection is established over the SNA network.

5 SO-d is prepared to receive data over a stream connection by issuing the *recv*. SO-s sends the connection-oriented data stream to SO-d. CMM-s sends the data stream as one record over the APPC conversation to CMM-d, where the data is received by SO-d.

Figure 66 on page 114 shows the process of sending a datagram over the SNA network.

1 The sockets application program SO-s sends a datagram using the sockets call *sendto* to the destination program SO-d. SO-d is prepared to receive connectionless data after it issues a *recvfrom*. (If a datagram arrives before the recvfrom is issued it is queued and delivered to the application once it issues a recvfrom.)

2 Sockets over SNA translates the destination IP address within the *sendto* to the the LU-d name. If no conversation to CMM-d is available, it issues an *allocate*. CMM-s adds an MPTN header to the datagram and sends it to CMM-d. There the datagram is presented to SO-d in the original form.

As in the case of connection establishment, if no session existed before, a session is established.

3 In this case, a conversation to CMM-d already exists, so the datagram is sent with the MPTN header to CMM-d and forwarded to SO-d.

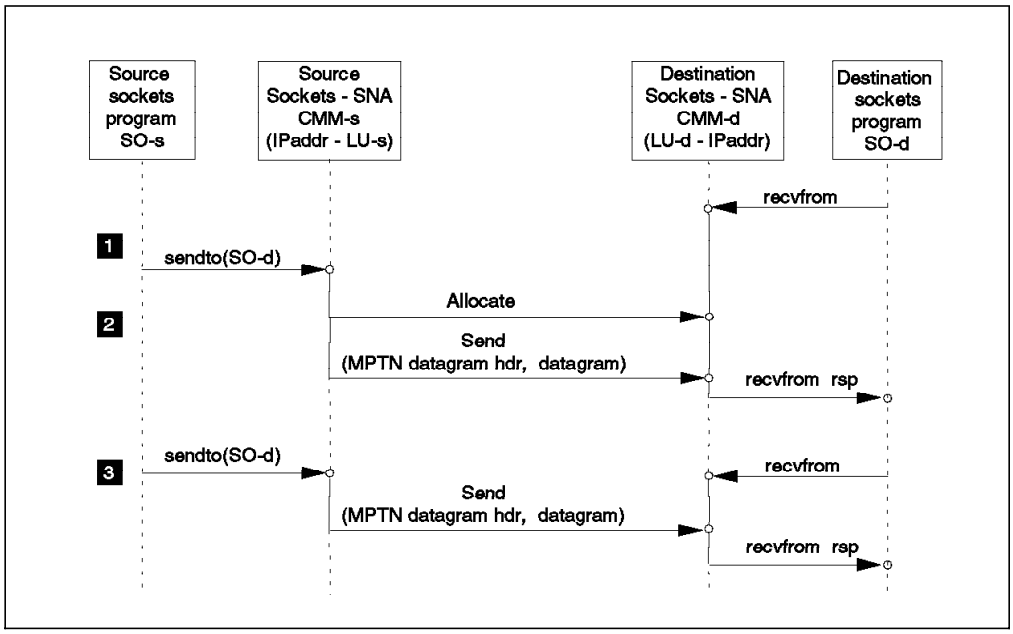


Figure 66. Connectionless Data Transfer in Sockets over SNA

7.2.10 Network Examples

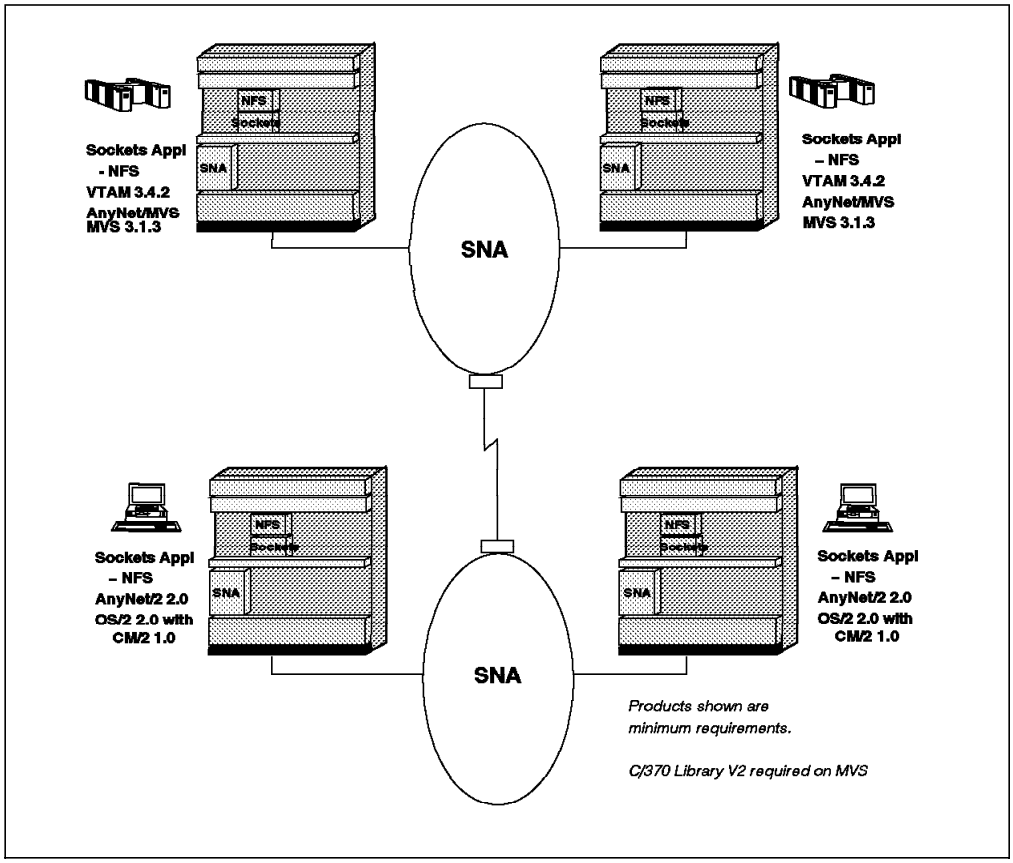


Figure 67. A Single Network with Sockets over SNA

Figure 67 shows one sample configuration using the AnyNet Sockets over SNA function. In this example, Network File System (NFS), a sockets application, is being used to communicate over an SNA network in client/server, workstation-to-host and workstation-to-workstation scenarios (in MVS, only the NFS server function is implemented).

Also shown are the minimum programming requirements for both MVS/ESA host and OS/2 workstation nodes.

Not all the MVS TCP/IP applications will currently operate over SNA. This is because most of these applications (like FTP and TELNET) are written to the Pascal interface to TCP/IP provided by IBM TCP/IP, instead of the sockets interface. Work is under way to get the MVS TCP/IP applications lined up to be converted to a C-language sockets interface.

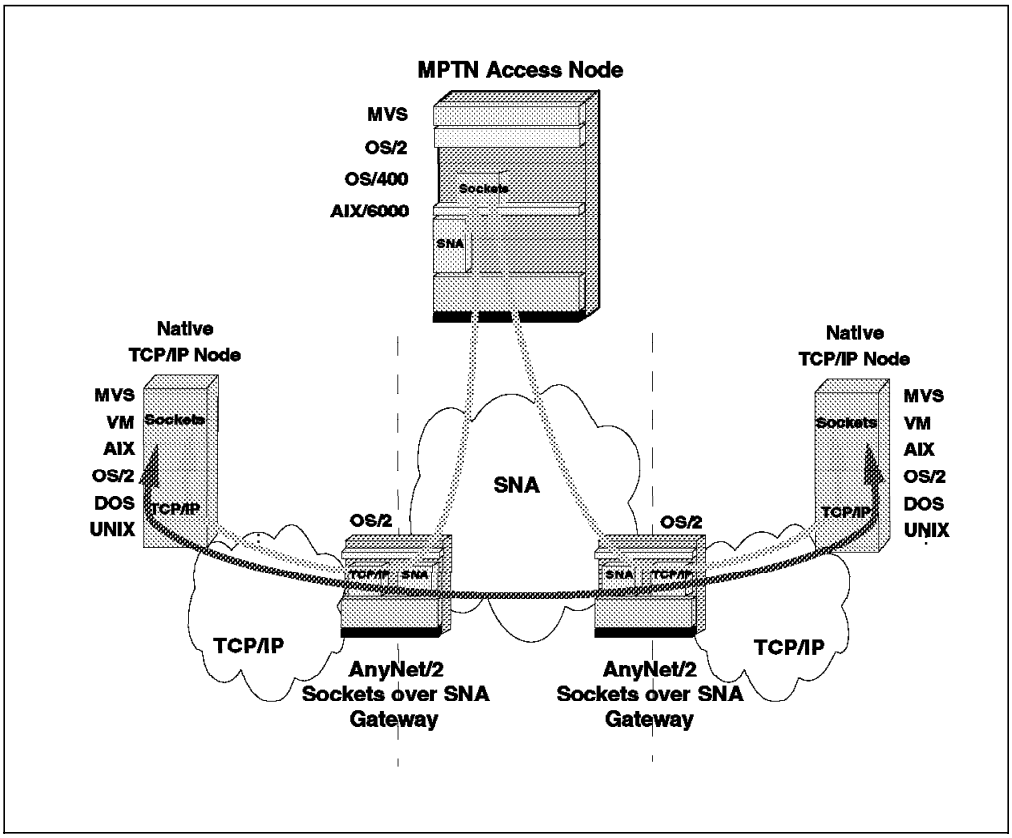


Figure 68. SNA and IP Networks Connected through Sockets over SNA Gateway

Figure 68 shows a possible configuration using the sockets over SNA transport gateway. A sockets application running in an MPTN access node communicates with matching applications in native TCP/IP nodes. The native TCP/IP nodes are not restricted to IBM hardware and software offerings, but can be any system that supports sockets applications and the TCP/IP protocol stack.

The sample network also shows that two AnyNet sockets over SNA gateways can be used to provide connectivity and data transport for sockets applications in native TCP/IP nodes through a backbone SNA network. The (TCP/IP) application nodes need not implement any MPTN or SNA functions; they are not even aware that an intermediate SNA network is used to transport their data traffic.

7.3 NetBEUI over SNA

A product based on the OS/2 platform is available which implements the NetBIOS Extended User Interface, NetBEUI, over SNA protocol combination as an MPTN access node.

Note: There is also a product which offers the gateway implementation of NetBIOS over SNA. This is the LAN-to-LAN-over-WAN[2] product which also runs on the OS/2 platform.

NetBIOS is designed as a communications protocol for local area networks, LANs, with low overhead appropriate for a medium offering broadcast communication. On a node supporting NetBIOS, the identification of the LAN adapter is the only routing function used, although the LAN adapter identification is logical. NetBEUI over SNA functions by providing another virtual LAN adapter. Communication with another such virtual LAN adapter on another node supporting NetBEUI over SNA uses APPC conversations through CM/2.

An important feature of NetBEUI over SNA is the use of a a workstation with a specialized function in support of NetBEUI over SNA, the *central services station*. The central services station provides the following functions:

1. the translation of 16-character NetBIOS names, the addressing scheme in NetBIOS communication, to SNA fully-qualified names
2. the maintenance of a directory database and its preservation on nonvolatile media, the “backup” function
3. the verification of NetBIOS names to try to preserve the uniqueness of these names in the set of communicating NetBIOS nodes
4. the management of datagram multicast operations

The name server function of the central services station operates in a manner very similar to the MPTN address mapping service as described in 6.3.1, “MPTN Address Mapping Service” on page 81. When a NetBEUI application starts, NetBEUI over SNA registers the application name with the name server function of the central services station. It is stored with the SNA LU name of NetBEUI over SNA. The in-storage data is written to a non-volatile medium periodically according to customization parameters.

This is all that happens with a *listening* or *server* program. A *calling* or *client* program will cause NetBEUI over SNA, in addition to registering the application name, to issue a locate request to the name server function of the central services station. This specifies the NetBIOS name of the application with which a connection is desired or to which a datagram must be sent. The name server function of the central services station returns the SNA LU name of the destination NetBEUI over SNA.

A locate function involves access to the name server function of central services station only if the association between transport user, NetBEUI, and transport provider, SNA, addresses has not already been cached by the NetBEUI over SNA implementation in the node. Such a cached entry is deleted if it becomes too

[2] *WAN*, wide area network, meaning, in fact, SNA, which is by no means limited to WANs. However, the intended use of the product is to offer “wide area” communication between LANs through the use the wide range of data link media support offered by CM/2.

old and the cache space is needed by a new entry. Also an entry is deleted if a connection attempt to the destination fails.

A compensation is needed to support multicasting and broadcast capabilities of NetBIOS over SNA. This is also a service offered by the central services station. The register process can also be to define a group name in addition to an application name. If a datagram is sent to a group name, it is sent to the central servives station and the central services station will send the datagram to all workstations with the same group name.

In the APPN network of interconnected machines running CM/2 with NetBEUI over SNA, it is recommended that the central services station be an APPN network node, while, typically, the end-user workstations and APPN end nodes.

The APPC functions which are of particular value to NetBEUI over SNA, as well as many other users of APPC, are as follows:

- Full-duplex
- Non-blocking calls
- Compression
- Security

The applications supported by NetBIOS which operate with NetBEUI over SNA include the following:

- IBM OS/2 LAN Requestor 3.0
- IBM OS/2 LAN Server 3.0
- Lotus Notes Release 3.0
- IBM Time and Place/2 Version 2.0
- IBM Person to Person/2 Version 1.0
- IBM NetFinity Version 1.1
- Service Installable File System, SRVIFS, used with C/I/D

The *routing preference table* concept of MPTN architecture is implemented in NetBEUI over SNA by means of the sequence in which an application, such as LAN Requester, tries to use the virtual LAN adapters. Thus if virtual LAN adapter 0 is set to use native NetBIOS and virtual LAN adapter 1 is set to use NetBEUI over SNA, virtual LAN adapter 0 is first tried in an attempt to find a partner and, if this was unsuccessful, virtual LAN adapter 1 is tried.

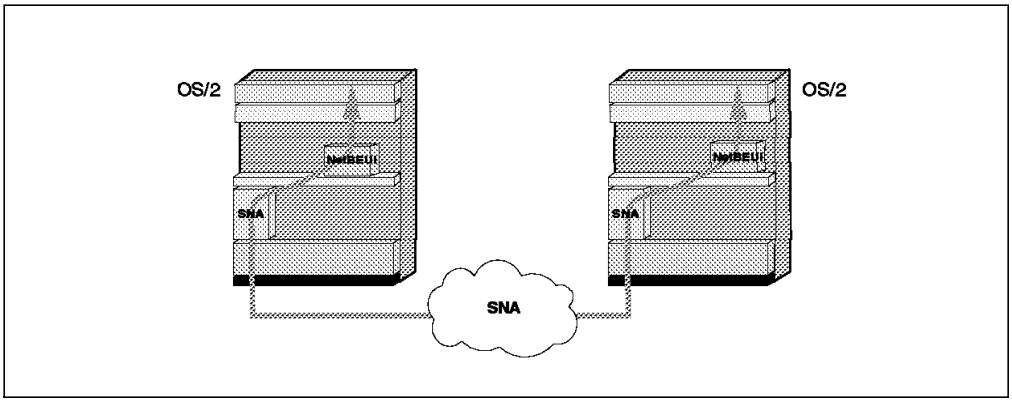


Figure 69. NetBEUI over SNA

7.4 MPTS - Sockets over NetBIOS

The Multiprotocol Transport Services (MPTS) - AnyNet for OS/2 function is incorporated into IBM Distributed Computing Environment (DCE) for OS/2. It allows TCP/IP sockets and NetBEUI applications to continue to communicate over TCP/IP and NetBIOS networks respectively. In addition, it is now also possible for TCP/IP applications such as FTP, TELNET, and NFS to communicate using NetBIOS protocols.

MPTS, like AnyNet/2, is based on the MPTN architecture and is a member of the AnyNet product family which provide multiprotocol combinations. Currently, either MPTS or AnyNet/2 may be run on one OS/2 system, but these two functions will be able to coexist on OS/2 with the next release of the AnyNet/2 sockets over SNA function.

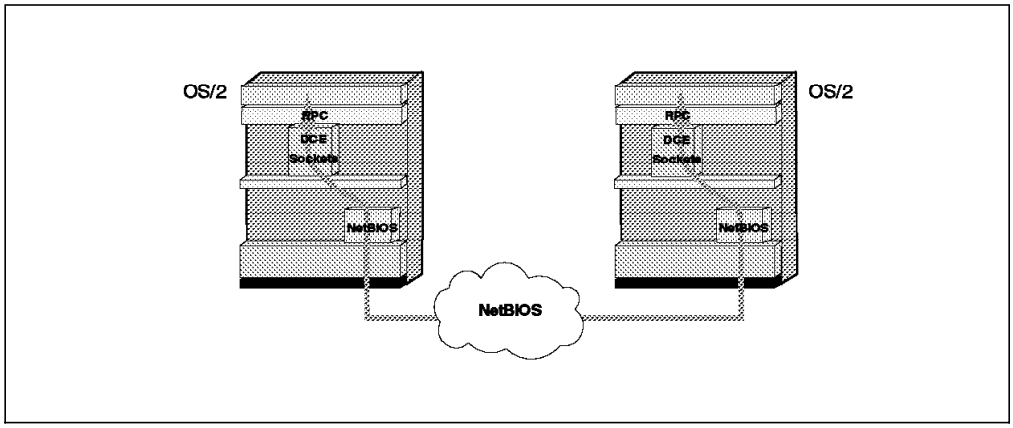


Figure 70. MPTS - DCE over NetBIOS

7.5 MPTN - SNA (APPC) over TCP/IP Customization Topics

7.5.1 Common

7.5.1.1 Address Translation

This topic was covered in the discussion about address mapping. In essence, the SNA fully-qualified LU name, the transport user address for SNA, is converted into a name which can be used with an IP network domain name server. This name is then translated, by reference to the domain name server, into an IP address, the transport provider address for TCP/IP.

The same process can also be performed using files local to the node. In the case of the MVS platform, which format these files have depends on whether the associated TCP/IP product is also present in the node. In the general case, the BSD format files, typically present in the ETC directory, are used.

In all cases the domain name suffix to be associated with the converted fully-qualified LU names must be defined.

In MVS this is the DNSUFX operand of the TCP major node VBUILD statement.

In OS/2 this is the SNASUFFIX environment variable.

7.5.1.2 Routing Preference Table

How this functions depends strictly upon the platform.

MVS: When a connection is to be established to a destination LU, the ALSLIST parameter of the destination LU's CDRSC statement defines if the (native) SNA network or the (nonnative) TCP/IP network is used for the session. If both connections are possible the order of the respective link station entries determines which network is tried first.

OS/2: A file LUTAB_LST defines the routing preference table for each fully-qualified destination LU name, although a default entry may also be specified. For each entry, this specifies one of the following:

- SNA first (native) and then TCP/IP (nonnative)
- TCP/IP first (nonnative) and then SNA (native)
- SNA only (native)
- TCP/IP only (nonnative)

The *lulist* command or the *lutpm* application tool may be used to maintain the routing preference table.

7.5.1.3 MPTN Parameter Customization

The following specifications are used to control or modify the timing parameters governing the operation of MPTN architecture for SNA over TCP/IP:

- The time to wait for a TCP connection
- The time delay before an SNA expedited request is also sent as a UDP datagram
- A timer to control
  - The interval between sending UDP datagrams for expedited request on an otherwise TCP connection
  - The time to wait for the closing of a TCP connection
- The interval between *keepalive* messages on a TCP connection

For testing purposes, it is possible to specify a well-known port for the SNA over TCP/IP product other than the standard, 397.

7.5.2 SNA over TCP/IP for MVS

Recall that the implementation of SNA over TCP/IP access node in the MVS is, in reality, a special case of an MPTN gateway node where the SNA network flow is internal to VTAM.

The central definition for SNA over TCP/IP for MVS is the TCP major node which has the structure of any definition for a link to a Type 2.1 node, namely, GROUP, LINE and PU statements.

The parameters given earlier and additional parameters specific only to the MVS platform are operands of the VBUILD statement of the VTAM TCP major node.

Additional considerations with SNA over TCP/IP on the over MVS platform are:

- the extent to which the TCP/IP for MVS product control blocks may need to be increased
- the number of tasks used by VTAM to interface with the TCP/IP for MVS product
- lastly, the connection between VTAM and the TCP/IP product is enabled by specification of the name of the address space running the TCP/IP product

Commands available to monitor the operation of the SNA over TCP/IP product in VTAM are as follows:

DISPLAY ID for the TCP major node

DISPLAY SESSIONS for a session supporting connections over the TCP/IP network

DISPLAY STATS for a summary of activities involving SNA over TCP/IP

In addition, some unsolicited messages will be generated to describe problems in the operation of SNA over TCP/IP. The NetView automation table could be used to generate alerts from such messages or as an entry point to further automated operation.

The need for CDRSCs, and their use to implement the routing preference table concept, has already been discussed. The use of dependent LU requester and server, DLUR and DLUS, in order to provide support for dependent LUs with the SNA over TCP/IP products on MVS and OS/2 has also already been mentioned.

7.5.3 SNA over TCP/IP for OS/2

What follows applies to the product which implements the MPTN access node support. It is also largely valid for the product which implements the gateway node support. Additional considerations for gateway node support are given in the next section.

SNA over TCP/IP on the OS/2 platform can operate as an extension of the TCP/IP product on the OS/2 platform. Thus, where the TCP/IP product is also installed, SNA over TCP/IP merely appears to implement another interface, *sna0*.

A consequence of this is that the same commands which are used to customize and operate TCP/IP on the OS/2 platform are also used to customize and operate SNA over TCP/IP. These commands are *ifconfig*, *route*, *netstat* and *ping*.

The eventual sessions flowing through the SNA network support LU 6.2 conversations through CM/2. Thus, SNA over TCP/IP is operated as an application supported by CM/2 and the panels available from the *subsystem management* icon together with the *aping* program, may be used to manage the SNA operation of SNA over TCP/IP.

The parameters given earlier and additional parameters specific only to the OS/2 platform are specified as *environment variables*.

Additional environment variables are as follows:

- How long to wait for the initiation of the TCP/IP product once the SNA over TCP/IP product has started
- How long to keep retrying to make a connection to a remote node when multiple alternatives have been defined by, say, entries in the domain name server

Other than the commands used in common with the TCP/IP product, the following commands are also avaialable:

abinfo a2b sna to show the association between SNA, transport user, and IP, transport provider, addresses for a particular session identified by the 16-hexadecimal character session id

abinfo b2a inet to show the association between IP, transport provider, and SNA, transport user, addresses for a particular IP address

7.5.4 SNA over TCP/IP for OS/2 Gateway

Additional considerations for MPTN gateway operation are as follows:

- A node can perform the MPTN access function and the MPTN gateway function at the same time.
- A node performing the MPTN gateway function must be defined in CM/2 as an APPN *network node* since it will perform the intermediate session routing function of APPN.
- The routing preference table definitions must always specify “try SNA first” or “try SNA only.”
- Gateway nodes may be accessed in parallel if the domain name server specifies more than one IP address for the name, specified in the domain name server, derived from the destination LU name.
- There may be a possibility to use wild-card definitions in the domain name server. This depends on being able to so name the LUs on the different sides of the gateway node that they have different net IDs. In comparison with APPN wild-card definitions, the domain name server wild-card definitions must be “full” wild card definitions rather than “partial” wild card definitions.
- The session limit associated with the gateway installation is defined as an environment variable.

7.6 MPTN - Sockets over SNA Customization Topics

7.6.1 Address Translation

This topic was covered in the discussion about address mapping. Because the use of fully-qualified LU names offers an address space that is larger than the IP address space, it is possible to map the IP address to a fully-qualified LU name with some considerable flexibility.

It is also possible to provide one-to-one mapping of IP addresses to fully-qualified LU names. This choice will tend to be made where only a small number of IP interface addresses needs to be translated into fully-qualified LU names.

In the IP environment, the most general case is to need to translate a host name to one of the IP interface addresses on that host before the IP address is handled by sockets over SNA. If the system running sockets over SNA has access natively to a domain name server, this can be handled in the usual way for an IP network. If such a name server is accessible through sockets over SNA over the nonnative, SNA, network, this also may be used. This latter access to a domain name server would be needed for sockets over SNA on the MVS platform since only access through the nonnative, SNA, network is possible for MVS.

On both the OS/2 and MVS platforms, the *sxmap*[3] command is used to maintain the definitions for how IP addresses are to be translated to fully-qualified LU namkes and the reverse.

7.6.2 Routing Preference Table

How this functions depends strictly upon the platform.

MVS: In sockets over SNA for MVS, the SPTN, native, TCP/IP, or nonnative, SNA, is selected when the program is bound to the modules supporting the access to the transport layer, strictly in MPTN terms, the common transport semantics (CTS) layer. In MVS, this process is *linkage editing*. If the sockets program is linkage edited with the modules supplied by the TCP/IP product, the SPTN is the native, TCP/IP, transport layer. If the sockets program is linkage edited with the modules supplied by the sockets over SNA implementation in MVS, a VTAM feature, the SPTN is the nonnative, SNA, transport layer.

OS/2: In sockets over SNA for OS/2, the SPTN, native, TCP/IP, or nonnative, SNA, is selected according to the routing specifications which have been entered by command in OS/2. Thus the routing preference is expressed indirectly. If the IP address for the destination node selects the sna0 interface, then the IP packet is routed over the nonnative, SNA, SPTN. If the IP address for the destination node selects any other interface, then the IP packet is routed over the native, TCP/IP, SPTN.

Failure of the interface normally selected can lead to selection of another interface and so, again, indirectly, it is possible to speak of “preference” in the MPTN routing process.

What has just been described assumes that both the sockets over SNA and the TCP/IP product have been installed in OS/2. If only the sockets over SNA product is installed, only access to the nonnative, SNA, SPTN is possible.

---

[3] In sockets over SNA for MVS, there are a number of commands which are supplied as load modules beginning with the principle module identifier for VTAM, IST. For example the sxmap command is, in fact, the ISTSKMAP module from SYS1.VTAMLIB. Such commands are shown as batch jobs to be run, typically, after the sockets over SNA address space has been started but before the sockets over SNA address space may be used. It is recommended that such command batch jobs be defined as cataloged procedures and the staged starting or running of the necessary procedures be performed under control of NetView automation.

7.6.3 MPTN Parameter Customization

In both the MVS and OS/2 platforms, customization parameters are specified as *environment variables*. This concept is well-known in OS/2. For MVS, this is a parameter data set or member available to the sockets over SNA address space.

The most significant common specification is the ability to select the SNA mode name and, hence, the performance characteristics of the path used through the SNA network. This is done on the basis of the TCP or UDP well-known port specified in the TCP or UDP header, in the case of TCP, when the connection is established. This topic was already discussed.

Other environment variables common to both the MVS and OS/2 implementations are as follows:

- The size of the buffer used to send SNA data
- The time to retain a converstion used to send datagrams before the conversation is deallocated following the sending of a datagram
- The technique used to segment datagrams to allow for compatibility with an earlier implementaion of sockets over SNA (AnyNet/2 V1.0)

7.6.4 Commands

As already mentioned, the commands used in both the MVS and OS/2 platforms have been standardized and, except for the sxmap command, have been standardised on the commands normally used with TCP/IP implementations, namely, *ifconfig*, *route*, *netstat* and *ping*.

7.7 MPTN Directions

When the first AnyNet products were announced, implementing MPTN access nodes for the APPC over TCP/IP and sockets over SNA functions for the MVS and OS/2 platforms, there were a number of *statements of direction* for future combinations. These and subsequent statements of direction have now largely been fulfilled.

Any future enhancements to the AnyNet family products and future products for new protocol combinations or an extension of existing protocol combinations on platforms with the combination not yet supported will be according to IBM business judgement.

Chapter 8. MPTN Network Management

A multiprotocol network is necessarily an heterogeneous network and the problems of network management are those of managing a heterogeneous network. Particularly they introduce the problems involved in managing multiprotocol connections and datagrams. Errors are reported at the end-points to the application protocol based on failures that occur at the transmission protocol. MPTN incorporates the use of network management protocols from today's networks. This is illustrated in Figure 71.

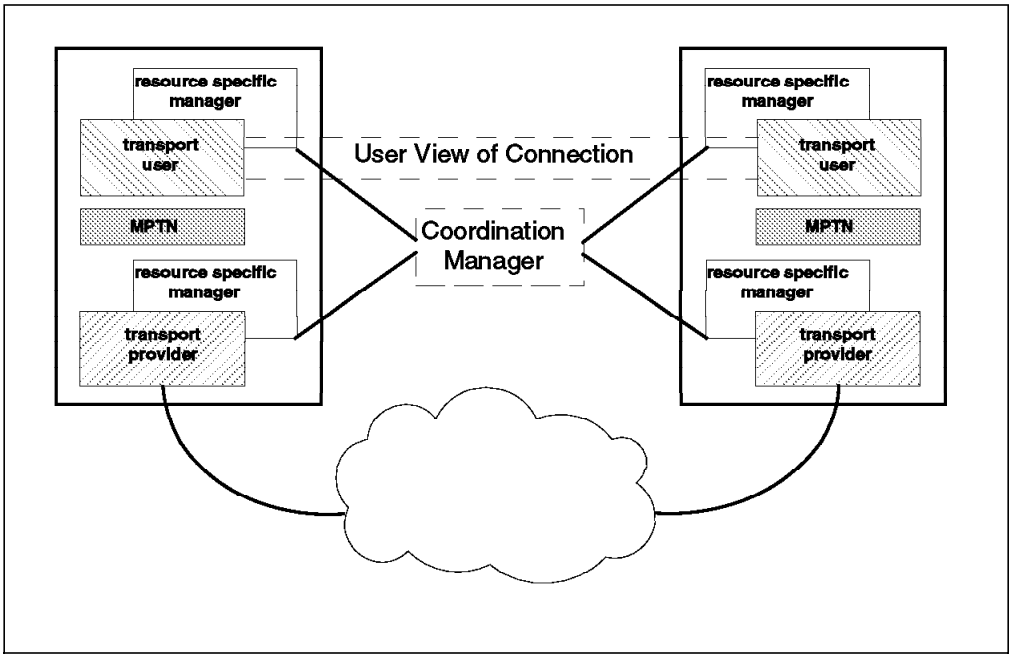


Figure 71. MPTN Network Management

Each environment will be managed natively and separately by Resource Specific Managers; the transport user and transport provider are managed natively and separately using existing management products. In an MPTN environment, resources may have their native management protocols such as SNA-Management Services (MS) for Alerts or Simple Network Management Protocol, SNMP, both of which are open architectures. Management of MPTN requires correlation among these different management protocols to identify problems that are signalled to a network management application as a result of incidents in the transport network.

Coordination Management identifies problems that appear at the application as a result of problems in the transport layer; MPTN conforms to the SystemView structure with Resource Specific Managers and Coordination Managers. SystemView Coordination Managers, such as NetView for MVS, consolidate information from SystemView Resource Specific Managers so the use of the various protocols can be consolidated.

MPTN components use SNA-MS Alerts to report errors. Transport protocol stacks are each managed natively by a Resource Specific Manager that understands the appropriate management protocol (for example, SNA-MS or SNMP). These Resource Specific Managers report to a Coordination Manager, such as NetView for MVS, typically using the Service Point Command Facility. In

some cases, there is translation of management protocols as in the case of NetView for AIX translating traps to alerts.

Investment in existing management products are protected. Investment in NetView for MVS, particularly automation, will be protected with consolidation of data at NetView for MVS and links to other management systems for automation and operator access.

8.1 APPC over TCP/IP Network Management

The APPC over TCP/IP MPTN function can utilize existing NetView for AIX management capabilities in order to manage APPC applications running over TCP/IP networks. The management of these APPC applications is essentially the well known SNMP capabilities already provided by NetView for AIX. NetView for AIX provides broad support for IP networks and all devices with IP connectivity running SNMP. One difference is that the SNA LU name associated with the APPC application needs to be correlated to its TCP/IP address in order to utilize the SNMP management in NetView for AIX.

Through the use of VTAM DISPLAY ID and DISPLAY SESSIONS commands or querying the domain name server, the SNA LU to TCP/IP address correlation can be made. Then, the SNMP capabilities are available in a TCP/IP network as usual. This is illustrated in Figure 72.

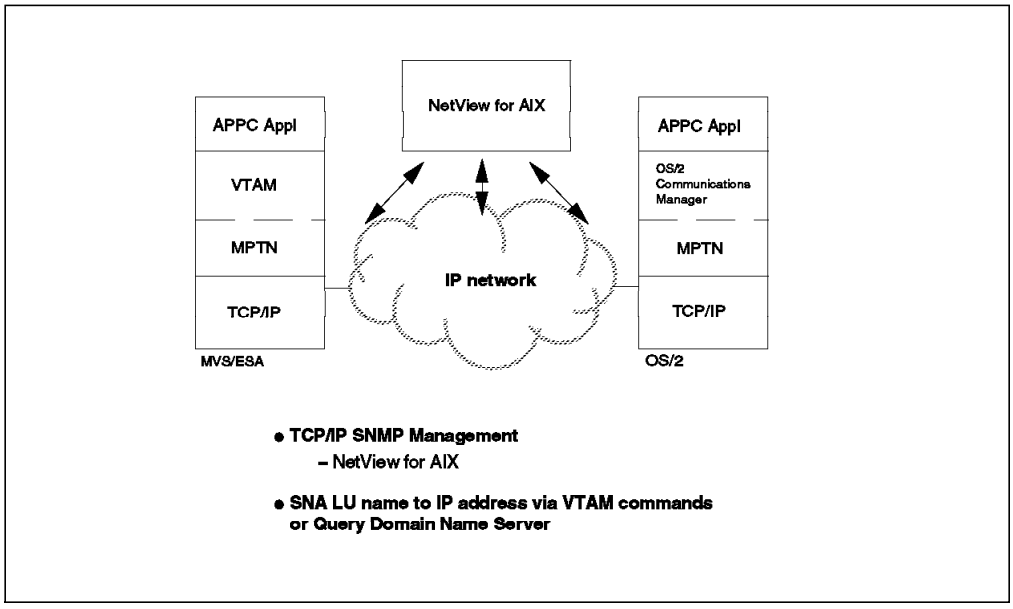


Figure 72. APPC over TCP/IP - Network Management

NetView for AIX provides generic support using MIB2 which allows for a significant number of IBM and non-IBM devices to be managed. AIX commands such as PING and TRACEROUTE give the user the ability to check connectivity and trace the route to a remote node.

If it is preferred to manage IP resources from NetView for MVS or to be able, perhaps at different times of the day, to manage IP resources from both NetView for AIX stations and NetView for MVS, the MultiSystem Manager product IP feature, working in conjunction with both NetView for MVS and NetView for AIX allows operations management on NetView Graphic Monitor Facility (NGMF)

displays. This allows a measure of integration with the management of networks running other protocols such as SNA APPN and subarea SNA from a centralized platform. Clearly this offers an opportunity for coordinated management of MPTN networks for IP and SNA-based protocol mixes. NGMF, through the use of Graphic Monitor Facility Host Subsystem (GMFHS) and the Resource Object Data Manager (RODM) provides topology and status displays for an expanding range of devices, SNA and non-SNA, depending upon the agents available for providing the topology and status data and the support for those agents within NetView for MVS. NGMF allows the user point and shoot capability for commands to the devices, again depending on the capability of the agents involved.

8.2 Sockets over SNA Network Management

The sockets over SNA function can utilize existing NetView for MVS management capabilities. One difference from business-as-usual NetView for MVS management is the fact that socket applications are associated with a port which has a TCP/IP address. The address mapping services provided with MPTN can be used to correlate the IP address to an SNA LU name. Once the SNA LU name is known, the network management of socket applications uses existing NetView for MVS capabilities. This is illustrated in Figure 73.

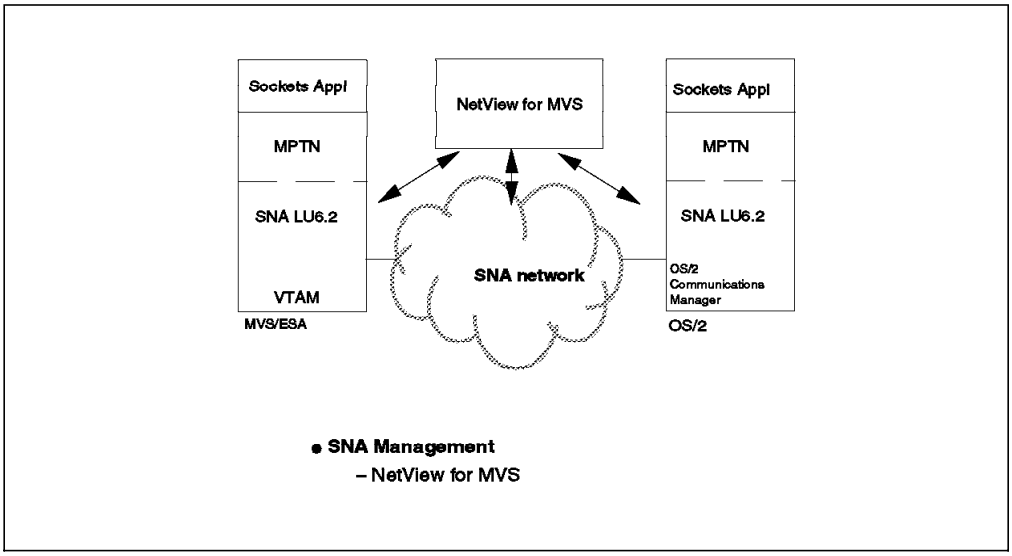


Figure 73. Sockets over SNA - Network Management

VTAM and OS/2 provide for assignment of LU names from IP names for management. Tracing of session routes through the SNA backbone to identify the failing node is also available.

The earlier comments concerning the management of SNA networks from NetView for MVS apply here also.

8.3 MPTN Management of both SNA and TCP/IP Networks

Through NetView for AIX and AIX NetView Service Point, SNMP traps are converted to alerts for display by NetView for MVS. The alerts can be customized at NetView for AIX. RUNCMD commands can be issued from NetView for MVS to AIX NetView Service Point for any AIX command, including SNMP and TCP/IP commands.

Additionally, when MVS TCP/IP is installed, NetView for MVS has access to SNMP commands through an MVS TCP/IP SNMP manager. NetView for MVS has access to MVS TCP/IP commands through TAF to TSO.

The NetView for MVS operator using NGMF can view the IP devices in the network with the use of the MultiSystem Manager IP feature and NetView for AIX.

Correlation of the alerts and traps from SNA and TCP/IP networks is enabled through the following functions:

- VTAM and OS/2 commands display IP addresses for sessions. (APPC over TCP/IP)
- TCP/IP commands recognize LU as an alias for IP addresses. (APPC over TCP/IP)
- LU names are derived from IP addresses. (Sockets over APPC)

8.4 VTAM MPTN Gateway Function Management

Gateway management uses NetView as the coordination manager to allow a comprehensive management solution through NetView for MVS and NetView for AIX synergy.

NetView for MVS is used for SNA management and NetView for AIX for the SNMP management. NetView for MVS can be used as a coordination point to manage both the SNA and TCP/IP networks. APPC connectivity is established via SNA session management. This allows use of the Session Manager, Route Test and Session display commands.

TCP/IP connectivity is established via TCP/IP commands. The use of PING and TRACEROUTE provides this. The LU names can be used to address IP devices for these commands.

Customization by means of the Graphic Monitor Facility Host Subsystem (GMFHS) and the Resource Object Data Manager (RODM) allows the creation of topology views which could incorporate nodes running dissimilar protocols in an MPTN network, among which the MPTN gateway nodes would be the most significant nodes to present. The support for the protocol mix, SNA and IP, would involve NetView for MVS, NetView for AIX and the MultiSystem Manager IP feature.

8.5 Future Directions

As for the future, MPTN introduces new concepts which need to be implemented:

- The association between the transport user and the transport provider in the MPTN Access Node
- The association between the native transport provider and the non-native transport provider connections at MPTN Gateway Nodes

The MPTN management goals for the future include evolving network management products to show the customer a single user interface, and providing network management consistency across protocols and systems.

Glossary

A

address mapper function. An MPTN component that maps nonnative transport-user addresses to a form used in native transport networks.

address space. The set of all legal addresses that may be formed according to the rules of a given address type. These rules include the maximum number of characters can be in the address and the permissible characters. Each protocol has its own set of rules. Since addresses in one protocol may be legal in another protocol, MPTN qualifies all names with an address type.

address type. An identifier in an MPTN header that indicates the protocol category (for example, OSI or TCP/IP) and hence the specific syntax and structure of the accompanying address. A given address plus its address type form an MPTN-qualified address.

API. Application programming interface.

application programming interface. An interface between the application program and the application support layer.

B

below-specific protocol boundary. The interface between the common MPTN manager (CMM) and the protocol-specific MPTN manager (PMM).

below-specific. Specific to one transport provider that exists below the CMM.

BSPB. Below-specific protocol boundary.

C

CMM. Common MPTN manager.

common MPTN manager. The component of the MPTN architecture that provides services independent of any protocol. The MPTN function that can be performed independent of any protocol. Examples include registering transport users with the MPTN address mapper component, selecting a transport provider, and establishing MPTN connections.

common transport semantics. The layer of the Networking Blueprint above the transport layer that makes the services of the transport provider available to the transport user.

compensation. The function of making up for differences in functions requested by the transport user and those provided by the transport provider.

composite network. A single-protocol transport network made up of multiple individual networks running the same transport protocol, each with its own unique net ID.

connectionless service. A service that treats each packet or datagram as a separate entity that contains the source and destination address. Connectionless services are on a best-effort basis and do not guarantee reliable or in-sequence delivery.

connection-oriented service. A service that establishes a logical connection between two partners for the duration that they want to communicate. Data transfer takes place in a reliable, sequenced manner.

CPI-C. Common Programming Interface for Communications.

CTS. Common transport semantics.

D

datagram. A self-contained packet, independent of other packets, that carries information sufficient for routing from the source transport user to the destination transport user.

datagram segment. A part of a datagram. A datagram may be segmented (that is, split into more than one part) if it contains too many bytes of data to send at one time.

DCE. Distributed Computing Environment.

DRDA. Distributed Relational Database Architecture of IBM.

E

expedited data. Data that is considered urgent. Such data may be delivered ahead of but no later than all normal data that preceded it.

F

FTAM. File transfer, access, and management.

FTP. File transfer program.

G

gateway-specific protocol boundary. The interface between the common MPTN manager (CMM) and the gateway protocol-specific MPTN manager (gateway PMM) and between the relay function and the gateway PMM.

group address. A single transport address identifying a collection of users. The collection of users is formed so that they can all receive common multicast datagrams.

GSPB. Gateway-specific protocol boundary.

H

heterogeneous network. A single logical network formed by MPTN gateways joining individual networks that support different network protocols.

I

IDRP. Inter-domain routing protocol.

intra-domain. An OSI term which refers to something within an OSI routing domain. When used in the term intra-domain routing, it qualifies the routing protocol as being used to route within a routing domain.

inter-domain. An OSI term which refers to something that is between OSI routing domains. When used in the term inter-domain routing, it qualifies the routing pro- tocol used to route between routing domains.

Inter-Domain Routing Protocol. The OSI inter-domain routing protocol. IDRP is a distance vector routing protocol that fixes the problems of past distance vector protocols (for example, looping).

IP. The network protocol that forms part of the Internet Protocol suite referred to as TCP/IP. The Internet Protocol defines the internet datagram as the unit of information passed across the internet, and provides the basis for the internet connectionless, best-effort packet delivery service.

IPX. Novell's Internetwork Packet Exchange protocol.

L

LU. Logical unit.

LU 6.2. An SNA logical unit that supports general communication between programs in a distributed processing environment.

M

matching. The relationship between peer transport users or peer transport providers that are of the same family.

MPTN. Multiprotocol transport networking.

MPTN access node. A node that has MPTN components installed, allowing transport users to use nonnative transport providers.

MPTN connection. An end-to-end connection through the MPTN network that may traverse multiple networks running different protocols. If the connection consists of multiple MPTN segments, it is formed having MPTN transport gateways concatenate the MPTN segments into one logical connection.

MPTN datagram. A datagram that carries an MPTN header as part of the data.

MPTN network. A network consisting of a mixture of native nodes, MPTN access nodes, MPTN address mapper nodes, and MPTN transport gateway nodes. The resulting network has the appearance to the user of one logical network. An MPTN network which consists of just a single SPTN does not contain an MPTN transport gateway.

MPTN-qualified transport address. A transport address that is qualified by its corresponding address type. The address conforms to the syntax and meaning of the specified address type. An example of an MPTN-qualified transport address is the pair (SNA, Netid.LUname).

MPTN transport gateway. A component of an MPTN network that connects two or more single-protocol transport networks.

MPTN segment. A connection across a single-protocol transport network between an MPTN node (either an MPTN access node or an MPTN transport gateway) and another node that may or may not be an MPTN node.

multicast. A technique that allows a single packet (or datagram) to be passed to a selected group of destinations that share a group address.

multicast datagram. A packet that is sent to more than one partner.

multiprotocol transport network. A collection of single-protocol transport networks, each using a different transport protocol.

multiprotocol transport networking (MPTN). The architecture for mixed-protocol networking and for network interconnection.

N

native. A relationship between a transport user and a transport provider that is based on the same transport protocol.

native network. With respect to a particular transport user, a transport network that provides the address type and transport characteristics assumed in the design of the transport user. No MPTN address mapping or compensation protocols are used for data transfer.

native MPTN segment. An MPTN segment that spans only a single protocol type and does not use any MPTN flows or headers. A native MPTN segment may form part of an MPTN connection.

native node. A node with no MPTN capability.

native routing gateway. With respect to a given transport network, a protocol-specific gateway that implements the same routing protocol as that network.

NetBEUI. NetBIOS Extended User Interface.

NetBIOS. Network Basic Input/Output System.

NetBIOS extended user interface (NetBEUI). The API to the NetBIOS transport protocol.

net ID. Network identifier. The address qualifier of a transport-user or transport-provider address that identifies a group of nodes according to the network in which they reside.

Network Basic Input/Output System (NetBIOS). A protocol used by many small computers for network access.

networking. Providing a relaying and routing service.

networking protocol. A specification of rules governing the exchange of information between components of a network.

nonnative. A relationship between a transport user and transport provider that are based on different transport protocols.

nonnative MPTN segment. A segment of an MPTN connection in which an MPTN protocol was used to establish the segment.

nonnative network. With respect to a particular transport user, a transport network whose addressing structure and transport service are different from that assumed in the design of that transport user.

nonnative protocols. Protocols used by a nonnative network.

nonnative transport address. A transport-user address having an address type different from the transport network address type.

NSAP address. That part of the address that uniquely identifies a network service entity, or network user.

NSAP. Network Service Access Point. A uniquely identified instance of the network provider. An NSAP is used to identify a network user on a certain node.

O

OSF. Open Software Foundation, Inc.

OSI. Open System Interconnection. The interconnection of open systems in accordance with the hierarchical arrangement of the seven layers of networking functionallity described in the specific International Standards Organization standards.

P

PMM. Protocol-specific MPTN manager.

protocol boundary. A generic description of a functional boundary defined by the architecture; implementations must conform to the semantics of the protocol boundary, but not necessarily to the syntax.

protocol-specific MPTN manager (PMM). A component of the MPTN architecture that performs management, routing, and binding functions that are performed differently for the different transport providers.

R

RDA. Remote Database Access of OSI.

record data format. A format that maintains record boundaries for the data being transmitted.

RFC. Request for Comment. The process by which some standards bodies define specialized solutions. In the case of MPTN, it is the definition of specialized transport protocols.

routing domain. A set of end systems and intermediate systems that operate according to the same routing procedures.

RPC. Remote procedure call.

S

SAP. Service access point.

service access point. The services of layer n to layer (n+1) are provided at the interface between the layer, at the points called n-service access points.

service mode. The designation by a transport user of the characteristics that must be maintained for a given connection or datagram transmission. Each network protocol has its own way of requesting these characteristics which must be mapped to the MPTN service mode.

single-protocol transport network. A collection of physically connected nodes that implement a single common transport protocol. A single protocol transport network may span multiple net IDs.

SNA. Systems Network Architecture.

SNI. SNA network interconnection.

SNMP. Simple network management protocol.

socket. The abstraction provided by Berkeley 4.3 BSD UNIX that allows an application program to access TCP/IP protocols.

SPTN. Single-protocol transport network.

stream data format. Data that has no record boundaries. The data is simply a stream of bits.

Systems Network Architecture (SNA). IBM′s communication networking architecture.

T

TCP. Transmission Control Protocol.

Transmission Control Protocol (TCP). The Internet standard transport-level protocol that provides the reliable, full-duplex, stream service for TCP applications.

TCP/IP. Abbreviation for the protocols (i.e., TCP, IP, UDP) that comprise the DARPA Internet protocol standards.

TLPB. Transport-layer protocol boundary.

transport characteristics. The set of transport services that a transport user expects, for example, whether data will be sent using connections or datagrams, and formatted as streams or records.

transport-layer protocol boundary. The MPTN protocol boundary that provides access in a protocol-independent fashion to multiple transport protocols.

transport network. An implementation of transport networking. Examples are parts of SNA, TCP/IP, OSI, IPX, NetBIOS, DECnet, and AppleTalk.

transport networking. The communication services provided at the transport layer and below.

transport provider. A component that provides the transport functions associated with a particular protocol stack.

transport-provider address. A transport address used to identify a transport provider.

transport service access point. The address that each transport layer user uses to access the services of the transport layer.

transport user. An application program or application support element that uses transport services to convey data through a network. A program that directly requests transport services.

transport-user address. A transport address used to identify a transport user.

TSAP. Transport service access point.

U

UDP. User Datagram Protocol.

unicast datagram. A packet that is sent to a single partner.

User Datagram Protocol (UDP). The TCP/IP protocol that allows an application program in one node to send a datagram to an application program in another node. UDP uses the internet protocol (IP) to deliver datagrams.

X

XTI. XTI (X/Open Transport Interface) is a specification that defines the X/Open transport service interface.

Index

D

E

F

G

H

I

L

M

N

O

T

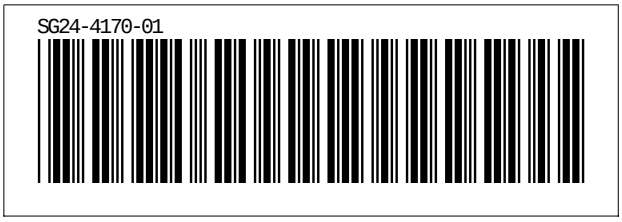
U

V

W

X

Printed in U.S.A.



Artwork Definitions

id	File	Page	References
ITSLOGO	MPTASU	i	i

Table Definitions

id	File	Page	References
TBL201A	MPTACH2	26	26, 26
TBL202A	MPTACH2	29	

Figures

id	File	Page	References
MPTA153	MPTACH1	5	1 4, 4, 4, 4
MPTA102	MPTACH1	6	2 5
MPTA151	MPTACH1	9	3 8
MPTA155	MPTACH1	9	4 9
MPTA152	MPTACH1	10	5 10, 10
MPTA154	MPTACH1	12	6 11, 12
MPTA201	MPTACH2	15	7 15
MPTA202	MPTACH2	16	8 16, 17, 17
MPTA203	MPTACH2	17	9 17
MPTA204	MPTACH2	20	10 17, 18, 19, 20
MPTA206	MPTACH2	21	11 18, 20
MPTA205	MPTACH2	22	12 17, 18, 19, 21
MPTA207	MPTACH2	23	13 22
MPTA208	MPTACH2	24	14 23
MPTA209	MPTACH2	24	15 24, 30
MPTA212	MPTACH2	25	16 25
MPTA213	MPTACH2	27	17 27
MPTA214	MPTACH2	28	18 27
MPTA215	MPTACH2	29	19 28
MPTA210	MPTACH2	30	20 30

MPTA211	MPTACH2	31	21	31, 32
MPTA409	MPTACH2	32	22	33
MPTA161	MPTACH2	33	23	34
MPTA162	MPTACH2	34	24	34
MPTA163	MPTACH2	35	25	35
MPTA165	MPTACH2	36	26	36
MPTA301	MPTACH3	40	27	39
MPTA310	MPTACH3	47	28	47
MPTA401	MPTACH4	53	29	53
MPTA408	MPTACH4	56	30	55, 81
MPTA406	MPTACH4	58	31	58, 59
MPTA302	MPTACH4	65	32	65
MPTA303	MPTACH4	65	33	66
MPTA304	MPTACH4	66	34	66, 66
MPTA305	MPTACH4	66	35	66, 66
MPTA501	MPTACH5	69	36	69, 69
MPTA502	MPTACH5	70	37	70
MPTA503	MPTACH5	71	38	71
MPTA508	MPTACH5	72	39	71, 72, 72
MPTA510	MPTACH5	74	40	73
MPTA690	MPTACH6	77	41	78
MPTA601	MPTACH6	79	42	79
MPTA650	MPTACH6	82	43	81
MPTA652	MPTACH6	86	44	85
MPTA654	MPTACH6	86	45	86
MPTA655	MPTACH6	87	46	86
MPTA691	MPTACH6	87	47	87
MPTA692	MPTACH6	88	48	89
MPTA602	MPTACH6	90	49	89

MPTA801	MPTACH7	92	50	92, 92
MPTA802	MPTACH7	93	51	
MPTA804	MPTACH7	94	52	94, 95
MPTA806	MPTACH7	96	53	96
MPTA851	MPTACH7	99	54	98
MPTA852	MPTACH7	100	55	99
MPTA853	MPTACH7	101	56	101
MPTA854	MPTACH7	101	57	101
MPTA855	MPTACH7	102	58	101
MPTA856	MPTACH7	102	59	102
MPTA803	MPTACH7	103	60	103
MPTA807	MPTACH7	104	61	104
MPTA808	MPTACH7	105	62	105
MPTA810	MPTACH7	107	63	106
MPTA812	MPTACH7	108	64	109
MPTA861	MPTACH7	113	65	112
MPTA862	MPTACH7	114	66	113
MPTA809	MPTACH7	114	67	115
MPTA816	MPTACH7	115	68	115
MPTA830	MPTACH7	117	69	
MPTA831	MPTACH7	118	70	
MPTA901	MPTACH8	125	71	125
MPTA902	MPTACH8	126	72	126
MPTA903	MPTACH8	127	73	127

Headings

id	File	Page	References
NOTICES	MPTAFM	xiii	Special Notices ii
BIBL	MPTAPREF	xvii	Related Publications
MPTA100	MPTACH1	1	Chapter 1, Introduction xv
WHYTRAN	MPTACH1	3	1.3, Why a Transport Solution? 11
MPTA102	MPTACH1	5	1.6, Positioning MPTN to Other Multiprotocol Solutions 2
MPTA101	MPTACH1	8	1.6.1, MPTN Internetworking
MPTA200	MPTACH2	15	Chapter 2, Architecture Overview xv
CH2FUN	MPTACH2	19	2.3.1, Basic MPTN Functions 19, 20
CH2CON1	MPTACH2	20	2.3.2, Configuration 1 - Single Nonnative Transport Network 19
CH2CON2	MPTACH2	20	2.3.3, Configuration 2 - Native-to-Nonnative Transport Networks (Gateway) 19
CH2CON3	MPTACH2	21	2.3.4, Configuration 3 - Dual Native Transport Networks (Gateway) 19
MPTA202	MPTACH2	22	2.4, Network Types and Connections 70, 70
MPTA201	MPTACH2	25	2.5, Address Formats and Mapping 18, 78
MPTA203	MPTACH2	30	2.6, Connection Establishment
INTRO22	MPTACH2	34	2.9.1.1, Merging Two Unlike Networks
INTROF3	MPTACH2	35	2.9.1.2, Departmental LAN Communication
MPTA300	MPTACH3	39	Chapter 3, MPTN Transport Services xv
MPTA309	MPTACH3	43	3.1.4, Selecting Transport Services
MPTA301	MPTACH3	45	3.1.5, Transport Protocol Characteristics for Specific Protocols 59
MPTA304	MPTACH3	46	3.2, Transport-Layer Protocol Boundary 40
MPTA400	MPTACH4	53	Chapter 4, MPTN Access Node xv, 46, 71, 73
MPTA402	MPTACH4	55	4.2, Transport Users
MPTA401	MPTACH4	59	4.4.2, The Protocol Compensator 40
MPTA404	MPTACH4	62	4.5, Connection Establishment and Datagram Routing
MPTA500	MPTACH5	69	Chapter 5, MPTN Gateway Node xvi
MPTA510	MPTACH5	73	5.2.1, The Gateway Protocol-Specific MPTN Manager
MPTA600	MPTACH6	77	Chapter 6, MPTN Address Mapping Services xvi, 48, 57
MPTA610	MPTACH6	79	6.2, MPTN Address Mapping Services Model
MPTA61A	MPTACH6	79	6.2.1, Registering Addresses
MPTA611	MPTACH6	80	6.2.1.1, Registering Individual Addresses
MPTA620	MPTACH6		

Index Entries

Footnotes

Revisions

id	File	Page	References
LE1	MPTAREV	i	xv, xv, xv, xv, xv, xv, xvi, xvi, xvi, xvi, xvi, xvi, xvi, xvi, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 15, 15, 16, 16, 16, 17, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 22, 23, 23, 23, 23, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 30, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 33, 34, 34, 34, 34, 34, 34, 34, 34, 34, 39, 39, 41, 41, 43, 43, 44, 44, 44, 44, 44, 44, 45, 45, 45, 45, 45, 45, 45, 45, 48, 48, 50, 50, 50, 50, 50, 50, 53, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54, 55, 55, 55, 55, 55, 55, 55, 55, 56, 56, 57, 57, 57, 57, 57, 57, 57, 57, 57, 57, 57, 57, 57, 58, 58, 58, 58, 58, 58, 58, 59, 59, 59, 59, 61, 61, 61, 61, 62, 62, 62, 62, 62, 62, 62, 62, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 64, 64, 65, 65, 65, 65, 70, 71, 72, 72, 72, 72, 73, 73, 73, 73, 74, 74, 74, 74, 74, 74, 74, 75, 75, 75, 75, 76, 76, 76, 77, 77, 77, 77, 77, 77, 77, 77, 77, 77, 78, 78, 78, 78, 78, 78, 78, 78, 78, 78, 78, 78, 78, 78, 79, 79, 79, 79, 79, 79, 79, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 82, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 86, 86, 86, 86, 86, 86, 86, 86, 86, 86, 86, 86, 88, 88, 88, 88, 88, 88, 88, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 90, 90, 90, 90, 90, 91, 92, 92, 92, 92, 92, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 93, 94, 94, 94, 94, 94, 94, 94, 94, 94, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 95, 96, 96, 96, 96, 96, 96, 96, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 101, 101, 101, 101, 101, 101, 103, 104, 104, 104, 104, 104, 104, 104, 104, 104, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 106, 106, 106, 106, 106, 106, 106, 106, 106, 107, 107, 107, 107, 107, 107, 107, 108, 108, 108, 108, 108, 108, 108, 108, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 110, 111, 111, 111, 111, 111, 111, 111, 111, 111, 111, 111, 112, 112, 112, 112, 112, 113, 113, 115, 115, 115, 117, 118, 121, 123, 123, 125, 125, 125, 125, 125, 125, 125, 125, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128
ED1	MPTAREV	i	xv, xv, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 11, 11, 11, 11, 15, 15, 15, 15, 16, 16, 17, 17, 17, 17, 18, 18, 18, 18, 19, 19, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26, 30, 30, 32, 32, 33, 33, 33, 33, 39, 39, 40, 40, 40, 40, 40, 40, 41, 41, 42, 42, 46, 46, 46, 46, 47, 47, 50, 50, 53, 53, 53, 53, 56, 56, 57, 57, 57, 58, 58, 58, 59, 59, 59, 59, 60, 60, 61, 61, 61, 61, 62, 62, 63, 63, 67, 67, 70, 70, 72, 72, 75, 75, 79, 79, 79, 79, 79, 79, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 81, 81, 82, 82, 82, 85, 85, 85, 87, 88, 88, 88, 88, 88, 88, 88, 88, 88, 89, 89, 89, 89, 89, 89, 95, 95, 95, 95, 95, 95, 95, 95, 97, 97, 98, 98, 115, 115, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125, 126, 126, 126, 126, 126, 126, 126, 127, 127

Spots

id	File	Page	References
PAGENO	SG244170 SCRIPT	133	(no text)

Tables

id	File	Page	References
TBL201	MPTACH2	26	1
			26
TBL202	MPTACH2	29	2
MPTA301	MPTACH3	45	3
			60
MPTA701	MPTACH7	91	4
			91
MPTA702	MPTACH7	91	5
			91
MPTA703	MPTACH7	91	6
			91
MPTA704	MPTACH7	112	7
			112

Processing Options

Runtime values:

- Document fileid ..... SG244170 SCRIPT
- Document type ..... USERDOC
- Document style ..... IBMXAGD
- Profile EDFPRF30
- Service Level 0029
- SCRIPT/VS Release 4.0.0
- Date 95.12.01
- Time 11:30:49
- Device 3820A
- Number of Passes 4
- Index ..... YES
- SYSVAR G ..... INLINE
- SYSVAR V ..... ITSCEVAL

Formatting values used:

- Annotation ..... NO
- Cross reference listing ..... YES
- Cross reference head prefix only ..... NO
- Dialog ..... LABEL
- Duplex ..... YES
- DVCF conditions file ..... (none)
- DVCF value 1 ..... (none)
- DVCF value 2 ..... (none)
- DVCF value 3 ..... (none)
- DVCF value 4 ..... (none)
- DVCF value 5 ..... (none)
- DVCF value 6 ..... (none)
- DVCF value 7 ..... (none)
- DVCF value 8 ..... (none)
- DVCF value 9 ..... (none)
- Explode ..... NO
- Figure list on new page ..... YES
- Figure/table number separation ..... YES
- Folio-by-chapter ..... NO
- Head 0 body text ..... Part
- Head 1 body text ..... Chapter
- Head 1 appendix text ..... Appendix
- Hyphenation ..... NO
- Justification ..... NO
- Language ..... ENGL
- Layout ..... OFF
- Leader dots ..... YES
- Master index ..... (none)

Partial TOC (maximum level)	4
Partial TOC (new page after)	INLINE
Print example id's	NO
Print cross reference page numbers	YES
Process value	(none)
Punctuation move characters	.,
Read cross-reference file	(none)
Running heading/footing rule	NONE
Show index entries	NO
Table of Contents (maximum level)	3
Table list on new page	YES
Title page (draft) alignment	RIGHT
Write cross-reference file	(none)

Imbed Trace

Page 0	MPTASU
Page 0	MPTAREV
Page 0	MPTAVARS
Page 0	MPTAFM
Page i	MPTAEDNO
Page ii	MPTAABST
Page xiii	MPTASPEC
Page xiii	MPTATMKS
Page xiv	MPTAPREF
Page xvii	MPTAWWW
Page xviii	MPTAACKS
Page xviii	MPTACH1
Page 13	MPTACH2
Page 37	MPTACH3
Page 52	MPTACH4
Page 67	MPTACH5
Page 76	MPTACH6
Page 90	MPTACH7
Page 123	MPTACH8
Page 128	MPTAGLOS
Page 139	MPTAEVAL